Symmetry and Complexity in Propositional Reasoning

Jim Molony

Ph.D.
University of Edinburgh
1999

Abstract

We establish computational complexity results for a number of simple problem formulations connecting group action and propositional formulas. The results are discussed in the context of complexity results arising from established work in the area of automated reasoning techniques which exploit symmetry.

Acknowledgements

My supervisors Alan Smaill and Stuart Anderson have provided consistent and valuable support in all aspects of the work. My thanks to them for allowing me great freedom to pursue my own ideas while always being ready to immerse themselves in whichever direction the work of the thesis happened to take.

I would also like to thank individually: Ursula Martin for much encouragement in looking for interesting properties connecting groups and propositional logic, and for sharing with me her insights into, and enthusiasm for, the subject; Roni Khardon and Mark Jerrum for very helpful suggestions of complexity issues to explore; Alex Hulpke for patient help with many questions on computational group theory; Andreas Schöter for supervisory advice in the early stages of the work.

I would add that all proofs, conjectures, observations etc. in the thesis are mine, except where otherwise stated, and any errors or inaccuracies in them are therefore also mine.

Acknowledgement is also due to the stimulating research environment of the Mathematical Reasoning Group at the Department of Artificial Intelligence, Edinburgh University, and to the excellent administrative and technical support infrastructure of the Department as a whole.

I would also like to acknowledge the financial support provided by the Edinburgh University Faculty of Science Scholarship which enabled me to carry out this research.

Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

Jim Molony
Edinburgh
September 20, 1999

Contents

List of Figures

Chapter 1

Introduction

This thesis is concerned with the computational complexity issues surrounding application of symmetry information in propositional reasoning. The well-known n-queens problems, place n queens on an $n \times n$ chessboard such that no queen attacks another, can be formulated as a satisfiability problem for a system of clauses expressing the constraints of the problem. The eight geometric symmetries of the problem translate into an invariance group of the set of clauses. Researchers have used example problems such as this to show that knowing a group of symmetries can give significant benefits in terms of reduced search spaces for various types of inference system. In fact, systematic application of symmetry techniques to problems such as n-queens can be traced back as far as [Gla74].

Using resolution as a base proof system for the propositional calculus, Krishnamurthy [Kri85] showed how certain tricky mathematical arguments could be encoded as short formal proofs in the resolution proof system augmented with a symmetry rule. It was observed by Boy de la Tour [BdlT96a] that in the approach of Krishnamurthy the principle of applying the symmetry knowledge is not contingent upon the symmetries being invariance groups of the set of clauses. He also considered the larger group of symmetries which fix the formula with respect to logical equivalence. In the 6-queens case, for example, we then obtain an extremely large group of permutations of the variables which preserve the formula with respect to logical equivalence: consider that the problem has just four models and that this in fact entails a large symmetry group.

In general, the larger the known symmetry group of the problem, the fewer distinct points there are to visit in the search space and the greater will be the benefits of the symmetry knowledge. On the other hand, computing with larger groups also adds a cost factor, but this has usually been found to be a good trade-off as the result of the polynomial complexity of many relevant permutation group algorithms.

This raises the question of how difficult it is to compute the group of logical, or semantic, symmetries of a propositional formula. Boy de la Tour showed that his problem GSYMM of computing a set of generators for the group of semantic symmetries is at least as hard as the satisfiability problem itself in the sense that an efficient computation of the semantic symmetries would imply an efficient algorithm for the satisfiability problem. This negative result would seem to indicate that there is little point expending effort to compute all the symmetries in order to improve the efficiency of checking satisfiability of a problem. However the proof of this result, and the question it raises over whether computing semantic symmetries could be useful, points to interesting connections between the action of groups on formulas and the complexity of problems such as satisfiability. For instance Boy de la Tour's proof exploits the fact that a formula is logically invariant under all[1] symmetries if and only if it is either valid or unsatisfiable. Alternatively and equivalently a formula which is not valid, for instance a nontrivial conjunctive normal form expression, is satisfiable if and only if there is some symmetry it does not have. The well-known satisfiability problem is thus rephrased as a symmetry problem.

1.1 Overview

The main contribution of this thesis is in establishing complexity results for a number of quite simple problem formulations connecting group action and propositional formulas. A glance at Table 8.1 illustrates some of the structure that emerges. A major theme of these investigations is how improved upper bounds on the complexity of these problems are obtained with certain restricted classes of formulas which might not initially be

[1] More precisely, at least a particular subgroup of the group W we define in Chapter 2.

suspected to exhibit such behaviour. A number of observations and conjectures are collected in a chapter following the main results. This chapter also presents summaries of resolved and unresolved special cases. In the process of arriving at these results we survey symmetry methods that have been applied in the literature and complexity results associated with them of relevance to this thesis. With the aim of providing a useful resource for further development of these results, we devote a chapter to transformation techniques used in the thesis, with the emphasis on the group-theoretic principles that link them. A puzzling element of tractability implied by the overall results for certain restricted classes of formula motivated some experimental work on a procedure called iterated Horn-closure. We describe the observations in a final chapter. In the following section we outline the structure and results of the thesis in more detail.

1.2 Thesis plan

Basic theoretical details of boolean terms, the propositional calculus, computational complexity, group and graph theory relevant to this thesis are included in **Chapter 2**. Some prior knowledge of complexity concepts and some elementary group theory are assumed although all definitions have been included to fix notation. This chapter also introduces the definitions of symmetry we use and of the main groups that we consider as transformations on boolean structures.

In **Chapter 3** we collect together five sets of tools for transforming problems. These provide a basis for both discussing the established results of Chapter 4 and for our results of Chapters 5–9. One of our main tools is a construction given by Martin connecting boolean functions $\mathcal{B}_n$ with systems of clauses of fixed size k [Mar98]. We also present in detail two constructions for manipulating formulas and groups necessary for exhibiting certain problem transformations. We summarise the relevant efficiency issues for manipulating finite permutation groups based on the stabiliser chain representation of [Sim70]. A number of useful techniques from the literature for transforming problems between graphs, sets and propositional formulas are reviewed.

In **Chapter 4** we survey some of the symmetry methods that have been developed by

researchers and review complexity results that have been associated with the methods previously, pointing out any results connected with work of the thesis. We also present a heuristic method that exploits structure of groups in two different search contexts. Experimental results show that group structure of the problem can be exploited independently of the problem itself.

To simplify the analysis of Chapters 5–8 we have structured them as follows. **Chapter 5** develops basic theorems and complexity analysis for the reduction tools that are used and fixes some conventions for the ensuing chapters. Chapters 6 and 7 present the group equivalence and containment problems and the main results for their complexity and inter-reducibility but without stopping to discuss the issues raised in any detail. Chapter 8 then takes up a number of the issues surrounding the complexity of these problems. In Chapter 5 we also review a result of Boy de la Tour's and give an alternative proof of his MSYMM problem which seems to improve the result and which proof provides a template for showing that two of the equivalence problems of Chapter 6 are essentially the same. All of the equivalence and containment problems of Chapters 6 and 7 are in the class Σ_2^p and we include in Chapter 5 a prototypical proof of membership for this class.

In **Chapter 6** we examine the *G-equivalence problem for boolean formulas* (G-EQUIV): given a pair of formulas $\phi(\vec{x}), \psi(\vec{x})$ and a group $G \leq W(\vec{x})$ determine the truth of

$$(\exists g \in G) \models \phi^g \leftrightarrow \psi$$

and the related *G-invariance problem* (G-INVAR) where we have to account for the fact that the identity element of a group always gives rise to a symmetry and so we consider only elements of G which are not the identity:

$$(\exists g \in G \setminus 1) \models \phi^g \leftrightarrow \phi.$$

We take G-EQUIV to be the most general of the problems of this chapter and pay particular attention to special classes of formulas of interest in automated reasoning, namely the k-CNF and k-DNF formulas. Relatively tractable cases include Horn formulas with bounded clause size and the case where clauses size is no more than two.

A third problem, the *C-invariance problem* asks the the question for a coset $C = Gk$ for some group G and permutation k

$$(\exists g \in C) \models \phi^g \leftrightarrow \phi.$$

We show this to be polynomially equivalent to the G-EQUIV problem, and it is the basic subroutine whose complexity can be used to calculate upper bounds on the difficulty of computing generators for the group of semantic symmetries for various classes of formulas. We also apply some previous results concerning equivalence testing of boolean functions represented as 'free boolean graphs' of which binary decision diagrams (Appendix A) can be viewed as special cases. These cases for G-EQUIV are also relatively tractable. We can encode graphs into binary decision diagrams to show for instance that testing G-equivalence for two binary decision diagrams is harder than the graph isomorphism problem but is still in NP. Questions concerning whether in fact G-EQUIV is Σ_2^p-complete and why it appears harder for CNF or DNF with unbounded clause size are deferred until Chapter 8 where similar issues involving the containment problems of Chapter 7 are also discussed.

In **Chapter 7** we examine the *G-embedding problem* for boolean formulas (G-EMBED): given formulas $\phi(\vec{x}), \psi(\vec{x})$ and group $G \leq W(\vec{x})$ determine the truth of

$$(\exists g \in G) \models \phi^g \rightarrow \psi$$

and we examine the *G-invariant subfunction problem* (G-ISF) which can be phrased as the problem of determining the satisfiability of

$$\bigwedge_{g \in G} \phi^g.$$

This is true if and only if there is a satisfiable formula ψ such that $\models \psi \rightarrow \phi$ and ψ is invariant (semantically) under G. These two problems are shown to be Σ_2^p-complete. We look at two further problems. The *$\mathcal{F}$-invariant subfunction* problem asks whether for formula ϕ and a family $\mathcal{F}$ of groups there is a member $G \in \mathcal{F}$ such that G, ϕ is an instance of G-ISF. This extra layer of quantification does not make the problem harder than G-ISF and we show it is Σ_2^p-complete. This problem corresponds closely

to a natural and general graph-theoretic problem, and we discuss the significance of a formal correspondence in Chapter 8.

A fourth problem, the C-disjunction problem (C-DISJ), is of interest because it is Σ_2^p-complete but has only one formula in the problem instance. As in Chapter 6 for the equivalence problems, we explore in particular for G-EMBED the complexity of the problem for particular special cases of formula, establishing an upper bound, for instance, for k-CNF formulas which is certainly better than for arbitrary CNF formulas according to current understanding of strict inclusion of complexity classes of the polynomial hierarchy.

In **Chapter 8** we discuss issues raised by the results of Chapters 6–7. We look at problem relating to the question of why CNF formulas with unbounded clause size seem harder than k-CNF for the problems of Chapters 6–7. We construct a number of pieces of evidence that G-EQUIV and related equivalence problems are not Σ_2^p-complete. We conjecture a correspondence between one of the Σ_2^p-complete containment problems and a natural graph-theoretic question.

Chapter 8 is intended to conclude most of the main points raised in the thesis. In **Chapter 9** we pursue from a different angle one of the more interesting points: the nice links between group action and k-CNF formulas. We define a tractable procedure called Horn-closure on 3-CNFs and show how it can be transformed into a randomised iterated procedure. We establish a number of symmetry arguments concerning the procedure and describe a number of experiments to locate hard problems for it.

Chapter 10 summarises the contributions of the thesis and points out the areas that appear to deserve further exploration.

The thesis has two brief **Appendices**, one on binary decision diagrams in the context of group action and the other on some of implementation details for Chapter 9. We also provide an **Index** of terms and keywords.

Chapter 2

Background Material

2.1 Introduction

In this chapter background and notational details are given for boolean terms, the propositional calculus, computational complexity, group and graph theory of relevance to this thesis. Some prior knowledge of complexity concepts and group theory are assumed. For complexity, the underlying notions of Turing machines, deterministic and nondeterministic computation and language recognition are needed. For groups, no advanced theory is needed in this thesis and we give all relevant definitions. However these are brief and without too much in the way of examples except for one or two less common constructions, and so some familiarity is assumed.

2.1.1 Plan of chapter

In §2.2 we fix some basic notation of general nature. The boolean terms for the propositional calculus are presented in §2.3 using terminology of Σ-algebras. We use these algebraic notions to define boolean terms and assignments to variables. The formality here is to emphasise and facilitate the description of properties of groups acting on the terms.

Reductions, completeness, standard complexity classes are summarised in §2.4. For the more familiar classes we give brief definitions. For the less familiar classes including the the classes of the polynomial hierarchy we expand a little more. Basic group and

permutation group definitions are given in §2.5. We also review some of the standard group-theoretic problems whose complexity is relevant to this thesis. Similarly for graphs, §2.6 gives basic definitions and some standard graph-theoretic problems which we will refer to later.

We define the action of groups on boolean functions and boolean terms in §2.7. Our primary concern is the action on terms since we deal with complexity problems for which term representations of boolean functions are used as part of the input language. Following the majority of workers in the symmetry area of automated reasoning we define and use a group W which is generated by both permutations of variables and complementation of variables. We collect in §2.8 a number of straightforward lemmas and observations about the group which will often be assumed without reference in later chapters. In §2.9 we look at the notions of the symmetry group of a boolean function or term, set out some routine properties and mention some more general issues concerning symmetry of relevance to later chapters.

2.2 General terminology

Sets The letters S, T, U denote arbitrary sets. We fix $\Omega = \{x, y, z, x_1, x_2, \ldots\}$ throughout the thesis as an infinite set of letters. We reserve V, X, Y to denote *finite* subsets of Ω. We use standard notation for set operations union, intersection etc. with $S \setminus T$ denoting set difference of S and T. Product and disjoint union are denoted by $\times, \oplus$. The relation $\subset$ denotes strict inclusion. The empty set is $\{\}$. The set T^k is the set of k-tuples over T and $T^{\{k\}}$ is the set of k-subsets over T. The powerset constructor $\mathcal{P}$ maps a set T to the set $\mathcal{P}(T)$ of sets over T. A finite sequence $x_1, x_2, \ldots, x_n$ of distinct elements of Ω is abbreviated by $\vec{x}$. A sequence $\vec{x}$ is collapsed to a set by $\{\vec{x}\}$. However we omit this extra notation within functions or predicates such as $\mathcal{P}, \subseteq$ understood to take sets as arguments.

Metalogic Theorems, lemmas, properties, etc. are stated using the metalogical symbols &, or, $\Rightarrow, \Leftrightarrow$, with $\not\models, \not\subseteq$ etc. denoting negation. Quantification is denoted by bracketed expressions such as $(\forall x \in X)$. The letter Q is sometimes used to range over $\{\exists, \forall\}$.

Big O notation We use the O notation as in $g(n) = O(f(n))$ to assert that g is of the order of f. If the converse also holds then we write $g(n) = \Theta(f(n))$.

2.3 Boolean terms and propositional calculus

In the following we develop the type of boolean terms as a Σ-algebra using notational conventions from [Hen89]. We define the boolean signature $B = \{\wedge, \vee, \neg, 1, 0\}$ with the usual arities for the connective symbols. We extend the signature B to a new signature $B(\Omega)$ where each $x \in \Omega$ is a new function symbol of arity 0. Hennessey's notation emphasises that Ω plays two different roles: viewed as a $B(\Omega)$-algebra, the term algebra $T_{B(\Omega)}$ is just the term structure over B with Ω included as additional generating elements; viewed as a B-algebra ($B \subseteq B(\Omega)$ and so any $B(\Omega)$-algebra is also a B-algebra), then the notions of substitution and assignment can be expressed in terms of B-homomorphisms with the set Ω taking the role of variables. The B-algebra $\mathcal{T}_B(\Omega)$ is then the algebra of *boolean terms* over variables Ω with constructor functions B (technically different from the symbols B but not notationally distinguished here).

Other connectives The other connectives we use are $\rightarrow, \leftrightarrow, +$ denoting implication, double implication and exclusive or. They are defined in terms of B by

$$\begin{aligned} \phi \rightarrow \psi &\stackrel{\text{def}}{=} \neg\phi \vee \psi \\ \phi \leftrightarrow \psi &\stackrel{\text{def}}{=} (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi) \\ \phi + \psi &\stackrel{\text{def}}{=} \neg\phi \leftrightarrow \psi. \end{aligned}$$

We use the precedence ordering $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, +$ to cut down on numbers of parentheses without loss of ambiguity. For example the term $(((\neg x) \wedge y) \rightarrow z) \leftrightarrow (\neg w)$ would be written equivalently $(\neg x \wedge y \rightarrow z) \leftrightarrow \neg w$.

Denote by 2 the set $\{0, 1\}$. Then there is a simple B-algebra $\mathbf{2} \stackrel{\text{def}}{=} \langle 2, B_2 \rangle$ where the operations over 0,1 are the standard truth-table rules for B. We use $b_1, b_2, \ldots$ to range over elements of 2.

2.3.1 Propositional calculus

In the nomenclature of the *propositional calculus* the boolean terms are called the *well-formed formulas* (wff) over the *propositional variables* Ω. We put $\mathsf{Wff} \stackrel{\text{def}}{=} \mathcal{T}_B(\Omega)$ and we will generally use this more accessible notation. Variables range over the *truth values* 0,1. The symbols B are called the *logical connectives*. The finite set of variables occurring in wff ϕ is denoted $\mathsf{Vars}(\phi)$. We also use the notation $\phi(\vec{x})$ to indicate that $\mathsf{Vars}(\phi) \subseteq \vec{x}$.

Assignments and Valuations An *assignment of truth values* to Ω is just a $\mathbf{2}$-assignment and we write Asg for the set of all $\mathbf{2}$-assignments to Ω. For any $a \in \mathsf{Asg}$ there is, by the Freeness Theorem for Σ-algebras, a unique B-homomorphism $a^{\#} : \mathsf{Wff} \to \mathbf{2}$ which agrees with a on Ω. The associated homomorphism $a^{\#}$ is called a *valuation* on Wff. We put $\phi(a) \stackrel{\text{def}}{=} a^{\#}(\phi)$ to minimise spurious notation. An assignment $a \in \mathsf{Asg}$ is called a *model* of wff ϕ if $\phi(a) = 1$.

We use the notation $\mathsf{Asg}(\vec{x})$ for finite $\vec{x} = x_1, \ldots, x_n \subset \Omega$ to represent the set 2^n of n-tuples of truth values, allowing us to quantify finitely over the relevant assignments in the context of wffs $\phi(\vec{x}), \psi(\vec{x}), \ldots$. Then we define the notion of valuation in the obvious way for $\mathsf{Asg}(\vec{x})$ by putting for wff $\phi(\vec{x})$ $\phi(\langle b_1, \ldots, b_n \rangle) \stackrel{\text{def}}{=} \phi(a)$ where a is some arbitrary $\mathbf{2}$-assignment such that $a : x_i \mapsto b_i$. Thus we have essentially just reorganised the set Asg in such a way that we can reason finitely about assignments with respect to a finite set of wffs.

Strings over 2 will sometimes be used to represent assignments $\mathsf{Asg}(\vec{x})$, with string $b_1 \ldots b_n$ corresponding to tuple $\langle b_1, \ldots, b_n \rangle$.

Truth relations Two wffs $\phi(\vec{x})$, $\psi(\vec{x})$ are *logically equivalent*, written $\phi \equiv \psi$, if $\phi(a) = \psi(a)$ for every $a \in \mathsf{Asg}(\vec{x})$. We say ϕ *logically implies* ψ, written $\phi \models \psi$ if $\psi(a) = 1$ whenever $\phi(a) = 1$. The wff $\phi(\vec{x})$ is *valid*, denoted $\models \phi$, if $\phi(a) = 1$ for every $a \in A(\vec{x})$, and *satisfiable* if there exists $a \in A(\vec{x})$ such that $\phi(a) = 1$. The equivalent expressions below on the right

$$\phi \models \psi \quad \Leftrightarrow \quad \models \phi \to \psi$$

$$\phi \equiv \psi \quad \Leftrightarrow \quad \models \phi \leftrightarrow \psi$$

will be the preferred formulation in this thesis for reasons of notational homogeneity.

2.3.2 Conjunctive and disjunctive normal form

Here we introduce some basic terminology. In §3.2 we look more closely at the properties of these normal forms with relevance to the complexity analysis of Chapters 6 and 7.

A *literal* is a wff of form x or $\neg x$ with $x \in \Omega$. Denote by Lit the set of literals over Ω. We let $\mathsf{Lit}(\vec{x})$ denote the finite set of literals over $\vec{x}$. A disjunction of literals is a term of form $l_1 \vee \ldots \vee l_n$ (strictly, with any choice of bracketing over $\vee$). A conjunction of disjunction of literals is a term of form $d_1 \wedge \ldots \wedge d_n$ where each d_i is a disjunction of literals. A wff in this form is said to be in *conjunctive normal form* (CNF). The dual version, a disjunction of conjunctions of literals, is called *disjunctive normal form* (DNF). We say a wff ϕ is in k-CNF (k-DNF) if each constituent disjunction (conjunction) contains k or fewer literals. The wff $\phi(\vec{x})$ is said to be in *full* CNF (DNF) with respect to $\vec{x} = x_1, \ldots, x_n$ if it is in n-CNF (n-DNF) and every constituent disjunction (conjunction) contains exactly n literals arising from the n distinct variables. A conjunction of literals in this case is referred to as a *minterm* .

2.3.3 Boolean functions

A boolean function is a map $F : 2^n \to 2$. We put $\mathcal{B}_n \stackrel{\text{def}}{=} 2^{2^n}$ for the set of boolean functions of n variables.

Recall that we identified 2^n with the assignments $\mathsf{Asg}(\vec{x})$ where $|\vec{x}| = n$. A wff $\phi(\vec{x})$ determines a boolean function $F_\phi : \mathsf{Asg}(\vec{x}) \to 2$ by $F_\phi : a \mapsto \phi(a)$. In relation to functions arising from wffs we sometimes speak of functions as subsets of $\mathsf{Asg}(\vec{x})$. We put $\mathsf{Mod}(\phi) \stackrel{\text{def}}{=} \{a \in \mathsf{Asg}(\vec{x}) \mid \phi(a) = 1\}$ to denote the set of models of ϕ with respect to $\vec{x}$.

The set B of connectives is *truth-functionally complete* with respect to any $\vec{x} =$

$x_1, \ldots, x_n$ and boolean functions $F \in \mathcal{B}_n$. That is, any function F is also the map $F_\phi \subseteq \mathsf{Asg}(\vec{x})$ for some wff $\phi(\vec{x})$.

2.4 Computational complexity

The main sources for complexity theory used in this thesis are [GJ79, Pap95]. The complexity classes we use are with respect to Turing machines as the underlying model of computation. The modes of computation we consider are probabilistic, deterministic and nondeterministic modes. The resource we wish to bound is time. In this section we briefly review some of the standard complexity classes relating to these parameters, also mentioning associated complexity problems relevant to later chapters.

2.4.1 Reductions

The concept of reduction captures what it means to say one problem is at least as hard as another. That is, problem A is at least as hard as problem B if B reduces to A. Many forms of reduction have been proposed. We utilise (or appeal to standard results which utilise) mainly the following three types of reduction: polynomial transformation; randomised polynomial transformation; Turing reduction. These are reviewed in this section.

Completeness The maximal elements of a complexity class $\mathcal{C}$ with respect to a reducibility are called $\mathcal{C}$-complete problems, and represent the 'hardest' problems of the associated class. If unqualified by any particular notion of reducibility, then completeness is understood to mean completeness with respect to polynomial transformation in this thesis.

Polynomial transformation A polynomial transformation from language L_1 to language L_2 is a function realisable by a polynomial time deterministic Turing machine (DTM) which takes as input an instance P_1 of L_1 and outputs an instance of P_2 of L_2 with the property that $P_1 \in L_1$ if and only if $P_2 \in L_2$. If such a transformation exists then we write $L_1 \leq_P L_2$. If $L_1 \leq_P L_2$ and $L_2 \leq_P L_1$ then we say the two languages

are polynomially equivalent, written $L_1 \equiv_P L_2$.

A language $L \in \mathcal{C}$ is $\mathcal{C}$-complete with respect to polynomial transformation if any language in $\mathcal{C}$ can be reduced to L by polynomial transformation.

Randomised polynomial transformation Reductions accomplished with the addition of randomisation to the polynomial time DTM have been used to establish completeness results for problems for which normal polynomial reductions have not been forthcoming (e.g. [VV85]). Language L_1 is reducible to language L_2 by a randomised polynomial time DTM, denoted $L_1 \leq_{RP} L_2$, if there is a randomised polynomial time DTM (i.e. equipped with a source of random bits) M which for any input instance $P_1 \in L_1$ outputs an instance $P_2 \in L_2$ with the property that: if $P_1 \in L_1$ then $P_2 \in L_2$ with probability at least a half; and if $P_1 \notin L_1$ then $P_2 \notin L_2$ with probability 1.

Turing reductions Turing (or 'Cook') reduction involves the notion of an oracle Turing machine. A Turing machine with an oracle for language A is a deterministic Turing machine, denoted M^A, equipped with a special 'query tape'. After writing a string s to this tape, the machine can invoke the oracle to decide $s \in A$ in a single computation step. Language L_1 Turing reduces to language L_2, denoted $L_1 \leq_T L_2$ if there is an oracle Turing machine M^{L_2} that can decide L_1 in polynomial time.

2.4.2 Standard time complexity classes

Complement The complement of a decision problem A is defined as the decision problem 'A COMPLEMENT' (for short names and acronyms A^c is used) whose answer is 'yes' whenever the answer to A is 'no' and vice versa. The complement of a complexity class $\mathcal{C}$ is the set of languages co-$\mathcal{C}$ $= \{\overline{L} | L \in C\}$. Note that all deterministic time classes are closed under complement.

The class P The class of languages decidable by a deterministic Turing machine in polynomial time is called P.

The classes NP and co-NP The class of languages decidable by a nondeterministic Turing machine (NDTM) in polynomial time is called NP. The abbreviations NPC and

co-NPC are used for NP-complete and co-NP-complete. The problem SAT for a wff $\phi(\vec{x})$ is the most well-known problem in NP:

Is ϕ satisfiable? Formally, is it true that $(\exists a \in A(\vec{x}))\ \phi(a) = 1$?

The problem SAT was proved NPC by Cook [Coo71]. We refer to SAT COMPLEMENT as the problem UNSAT for a wff ϕ:

Is ϕ unsatisfiable? Formally, is is true that $\models \neg\phi$?

Note that by putting $\neg\phi$ into the problem in place of ϕ this problem is essentially equivalent to the one which asks if ϕ is valid. It is an easy consequence of Cook's theorem that UNSAT is co-NPC.

The class DP A language L is in DP if and only if there are two languages $L_1 \in$ NP and $L_2 \in$ co-NP such that $L = L_1 \cap L_2$. The following problem is DP-complete and is not known to be in either of NP or co-NP. SAT/UNSAT asks for two wffs ϕ, ψ:

Is it the case that $\phi \in$ SAT and $\psi \in$ UNSAT?

DP stands for 'difference polynomial'. The class was introduced in [PY82].

The polynomial hierarchy The material here is adapted from [GJ79]. Recall the notion of an oracle machine M^A from the discussion of Turing reduction above. By appropriate generalisations of this notion, we define a hierarchy of complexity classes as follows. First consider the class P^A. This is defined to be the class of languages that can be decided in polynomial time with an oracle for language A. Similarly, NP^A is defined to be the class of languages that can be decided in nondeterministic polynomial time with an oracle for language A. Generalising further we define the classes $\mathrm{P}^{\mathcal{C}}$ and $\mathrm{NP}^{\mathcal{C}}$ as those decidable by the respective machines using any oracle from the class $\mathcal{C}$. We now have a method of defining the *polynomial hierarchy* (PH) recursively. Put $\Sigma_0^p = \Pi_0^p = \Delta_0^p = P$ and for all $k \geq 0$:

$$\Delta_{k+1}^p = \mathrm{P}^{\Sigma_k^p} \quad \Sigma_{k+1}^p = \mathrm{NP}^{\Sigma_k^p} \quad \Pi_k^p = \text{co-}\Sigma_k^p.$$

It is not so easy to check directly from this definition that a problem is in one of these classes and so we recall the standard membership criteria for the complexity classes Σ_k^p and Π_k^p.

Theorem 2.4.1 [Wra76] Let Γ be an alphabet with $|\Gamma| \geq 2$ and let $L \subseteq \Gamma^*$ be a language. For any $k \geq 1$, $L \in \Sigma_k^p$ iff there exists polynomials $p_1, \ldots, p_k$ and a polynomial time DTM recognisable relation $R \subseteq (\Gamma^*)^{k+1}$ such that for all $x \in \Gamma^*$

$$\begin{aligned} x \in L \quad \Leftrightarrow \quad & (\exists y_1 \in \Gamma^* \text{ with } |y_1| \leq p_1(|x|)) \\ & (\forall y_2 \in \Gamma^* \text{ with } |y_2| \leq p_2(|x|)) \\ & \vdots \\ & (Q y_k \in \Gamma^* \text{ with } |y_k| \leq p_k(|x|)) \\ & \langle x, y_1, \ldots, y_k \rangle \in R \end{aligned}$$

where the quantifiers alternate and where the final Q is $\exists$ if k is odd and $\forall$ if k is even. $L \in \Pi_k^p$ iff its complement $L^c \in \Sigma_k^p$.

We also recall the standard complete problems B_k for each Σ_k^p which were first identified in [MS72]. An instance of B_k consists of a wff $\phi(\vec{x_1}, \ldots, \vec{x_k})$ and the following question where $(Q\vec{x_i})$ is shorthand for $(Qx_{i_1} \in 2) \ldots (Qx_{i_n} \in 2)$

Is it true that $(\exists \vec{x_1})(\forall \vec{x_2}) \ldots (Q\vec{x_k}) \models \phi$?

where the quantifiers alternate and where Q is $\exists$ if k is odd and $\forall$ if k is even.

Theorem 2.4.2 [MS72, Wra76] For all $k \geq 1$, B_k is complete for Σ_k^p and the complementary problem B_k^c is complete for Π_k^p with respect to polynomial transformability.

Note that B_1 and B_1^c are respectively just SAT and UNSAT.

Summary of class inclusions Note that all of these containments are thought to be strict:

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{DP} \subseteq \Delta_2^p$$

$$\text{NP} = \Sigma_1^p \subseteq \Sigma_2^p \subseteq \cdots \subseteq \Sigma_k^p$$

$$\text{co-NP} = \Pi_1^p \subseteq \Pi_2^p \subseteq \cdots \subseteq \Pi_k^p$$

$$\Delta_k^p \subseteq \Sigma_k^p \cap \Pi_k^p.$$

2.4.3 Function versions of decision problems

Function versions of decision problems are problems for which we require the Turing machine to halt with more than a yes or no answer. In general, for a class $\mathcal{C}$ defined by a given type of Turing machine, then F-$\mathcal{C}$ is the class of functions from strings to strings that can be computed by a Turing machine of the same mode (deterministic, probabalisitc, etc.) and resource bound (time here). For example F-NP is the class of functions that can be computed by NDTMs; F-Δ_2^p is the class of functions that can be computed by a polynomial time DTM with a SAT oracle.

2.5 Groups

Sources for group theory used in this thesis are [DM96, LW96, But91]. A set G together with a binary operation $\circ$ called multiplication or composition forms a group if: (i) G is closed under $\circ$; (ii) the operation $\circ$ obeys the associativity law: $x \circ (y \circ z) = (x \circ y) \circ z$ for all $x, y, x \in G$; (iii) there exists an element $\epsilon \in G$ such that $x \circ \epsilon = x$ for all $x \in G$; (iv) for each $x \in G$ there is an x^{-1} in G such that $x \circ x^{-1} = \epsilon$.

Alternatively, a group G is an algebra with operations $\circ, \epsilon, {}^{-1}$ obeying at least the three equations: $x \circ (y \circ z) = (x \circ y) \circ z$, $x \circ \epsilon = x$, $x \circ x^{-1} = \epsilon$. From these three equations we can derive the other standard rules: $\epsilon \circ x = x$, $x \circ x^{-1} = \epsilon$, $\epsilon^{-1} = \epsilon$ etc.

Generators If $K \subseteq G$ then the group $\langle K \rangle$ denotes the smallest subgroup of G containing the set K.

Trivial group The trivial group consisting of just the identity is written as 1.

Order The order of a group G is the number of elements in it, possibly infinite. The order of an element $g \in G$ is n where n is the smallest positive number such that

$g^n = \epsilon$. If there is no such n then the order of g is infinite.

Permutation groups The set of bijections $g : T \to T$ on a set T forms a group $S(T)$ called the symmetric group on T with composition of maps as the group multiplication. We write gh for the composition of two elements of $S(T)$ and use ϵ for the identity mapping. With one or two infinite exceptions all the groups used in this thesis will be finite permutation groups. For these we use the standard cycle notation to represent permutations. For example let $\vec{x} = x_1, \ldots, x_5$ and then $(x_1\ x_2\ x_3)(x_4\ x_5)$ is an element of $S(\vec{x})$ of order 6 corresponding to the bijection $\{x_1 \mapsto x_2, x_2 \mapsto x_3, x_3 \mapsto x_1, x_4 \mapsto x_5, x_5 \mapsto x_4\}$. The symmetric group over $\vec{x} = x_1, \ldots, x_n$ is of order n and is generated conventionally by the following $n-1$ permutations

$$(x_1\ x_2), (x_1\ x_3), \ldots, (x_1\ x_n).$$

The *degree* of a finite permutation group $G \leq S(\vec{x})$ is the number $k \leq |\vec{x}|$ of points in $\vec{x}$ moved by G. The symmetric group over n arbitrary points is denoted by S_n.

Subgroups, cosets and transversals Let G be a group. For two subsets $A, B \subseteq G$ we use

$$AB \stackrel{\text{def}}{=} \{ab \mid a \in A, b \in B\}$$

to denote the set of elements formed by multiplying pairs of elements from A and B. In certain cases that will be pointed out, AB could be a group. A *subgroup* J of G is a subset of G which is also a group under the composition of G. When J is a subgroup of G we write $J \leq G$ or when J is known to be a strict subgroup then we put $J < G$. If $J \leq G$ and $g \in G$ then $Jg = \{jg \mid j \in J\}$ is called a *right coset* of G relative to J. Two right cosets Jg,Jh are either equal or disjoint and so G is partitioned into right cosets of equal size. A *right transversal* C of J in G is a set of representatives of the right cosets of J in G (one element from each distinct right coset). We use the notation $C = G : J$ to denote a right transversal of J in G. The *index* of J in G is the size of the transversal (also equal to the order of G divided by the order of J) and is denoted $[G : J]$. Right transversals are not unique but one convention we adhere to is that the identity ϵ is always selected as the representative of the identity coset $J\epsilon = J$.

Group action Let G be a group. G is said to act on a set T if there is a mapping from $G \times T$ to T (denoted by superscription) satisfying $x^\epsilon = x$ and $x^{gh} = (x^g)^h$ for all $x \in T$ and $g, h \in G$. The orbit of $x \in T$ under G is defined as $\mathsf{Orb}(x, G) = \{x^g \mid g \in G\}$.

A group G acts *transitively* on T if for each $x, y \in T$ there is an element $g \in G$ such that $x^g = y$. This is equivalent to saying $\mathsf{Orb}(x, G) = T$ for any $x \in T$.

Any subgroup $G \leq S(T)$ has of course a natural action on the set T. We frequently refer to *induced actions* of G on objects constructed from T such as the powerset $\mathcal{P}(T)$, the k-subsets $T^{\{k\}}$, product T^k or disjoint union $T \oplus \ldots \oplus T$, or some combination of these. For instance the action of G on T induces an action on $\mathcal{P}(T)$ by $s^g = \{x^g \mid x \in s\}$ or on T^k by $\langle x_1, \ldots, x_k \rangle^g = \langle x_1^g, \ldots, x_k^g \rangle$. If T are the vertices of a graph then the G action on T induces an action on edges by the natural action on T^2 for directed graphs, or by the natural action on $T^{\{2\}}$ for undirected graphs.

Induced actions of G on a set U give rise to a permutation group on U which we sometimes explicitly denote by $G[U] \leq S(U)$. For example if $U = T^2$ then $G[T^2]$ is the permutation group induced by G on ordered pairs of elements over T, that is the group of permutations $\{h_g : g \in G\}$ where $h_g : \langle x, y \rangle \mapsto \langle x^g, y^g \rangle$.

Wreath products Certain groups that we use or construct in this thesis can be viewed as wreath products of two permutation groups and we briefly review the construction in this section. Let T, U be two finite sets and let $G \leq S(T)$, $J \leq S(U)$. We define a group $W \leq S(T \times U)$ in terms of two component groups C, B as follows. Let C be the group induced on $T \times U$ by J acting on U, that is $j : \langle x, y \rangle \mapsto \langle x, y^j \rangle$. If we think of $T \times U$ as a matrix with entries $\langle x, y \rangle$ then C defines a group a permutations of whole rows of the matrix. Let B be the group induced on $T \times U$ by G acting on T *independently* in each row. In other words, B is the largest subgroup of $S(T \times U)$ which does not move elements between rows and where each permutation within a row corresponds to some $g \in G$ acting on the T components. The group generated by the union of B, C is the standard wreath product of G by J written $G \wr J$. The group B is called the *base* of the wreath product. Note that $B \cap C = 1$. Every element $w \in W$ can in fact be decomposed into a permutation bc with $b \in B$ and $c \in C$ and so we

can write $W = BC$. We have also $|W| = |B||C|$. See Example 2.6.1 below. These definitions extend naturally to the wreath product of infinite groups.

Setwise stabiliser Let $T \subseteq U$ and $G \leq S(U)$. The *setwise stabiliser* $\mathsf{Stab}(T, G)$ of T in G is the subgroup of G which fixes the set T, that is $\{g \in G \mid T^g = T\}$.

2.5.1 Group-theoretic problems

We recall a number of complexity problems concerning finite permutation groups. All the problems in this section are in NP and are polynomially equivalent to one another. See for example [Hof82, Luk82]. The induced action of $G \leq S(T)$ on $\mathcal{P}(T)$ gives rise to the ORBIT PROBLEM FOR SETS (OPS) for a pair $s, t \subseteq T$ and group $G \leq S(T)$:

> Are s, t equivalent under G? Formally, is it true that $(\exists g \in G)\ s^g = t$?

This problem is also known as the 'string isomorphism' or 'G-equivalence' problem for strings over 2 with G acting naturally on binary strings as representations of sets. GROUP FACTORISATION is the problem where we are given two groups $A, B \leq S(T)$ and an element $g \in S(T)$:

> Does g belong to AB?

COSET INTERSECTION has the same instances but asks:

> Is the intersection $Ag \cap B$ nonempty?

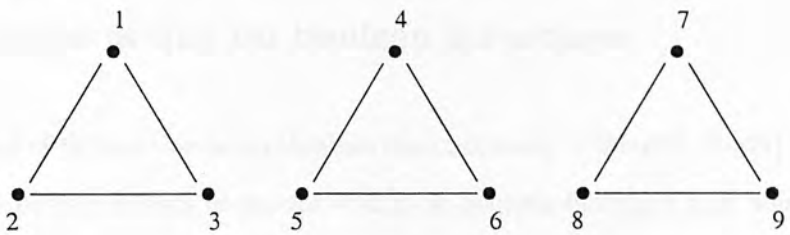
2.6 Graphs

References for graph theory used in this thesis are [Hof82, Bab95]. We tend to use H_1, H_2 for graphs to avoid confusion with groups G_1, G_2. A directed graph H is a pair $\langle V, E\rangle$ where $E \subseteq V^2$ is the set of edges. A typical edge is $\langle x, y\rangle \in E$. An undirected graph H is denoted by the pair (V, E) whose edges $(x, y) \in E \subseteq V^{\{2\}}$ are unordered pairs. The *complete graph* on V is the graph containing all V^2 or $V^{\{2\}}$ edges. A *cyclic*

graph over V is a graph consisting of $|V|$ edges in a single directed circuit or single undirected loop.

A graph is called *rigid* if it has no nontrivial automorphisms. The *automorphism group* of a directed or undirected graph H is the subgroup $\mathsf{Aut}(H) \leq S(V)$ of permutations of the vertices which fixes the set of edges according to the natural actions on directed or undirected edges.

Example 2.6.1 Consider the undirected graph H below.



The group $\mathsf{Aut}(H)$ of automorphisms of H can be described in terms of the wreath product $S_3 \wr S_3$. The two components C, B arise as follows. The group C is fixed as a group which just permutes the three triangles. The base B is the group which independently permutes the vertices of the triangles. Using the numbers 1,...,9 to represent $\{1,2,3\} \times \{1,2,3\}$ we can define C, B in concrete terms by

$$C \stackrel{\text{def}}{=} \langle (1\ 4)(3\ 6)(2\ 5), (1\ 7)(2\ 8)(3\ 9) \rangle$$

$$B \stackrel{\text{def}}{=} \langle (1\ 2), (1\ 3), (4\ 5), (4\ 6), (7\ 8), (7\ 9) \rangle.$$

C has order $3! = 6$ and B has order $(3!)^3 = 216$. Therefore $\mathsf{Aut}(H) = BC \cong S_3 \wr S_3$ has order $6 \times 216 = 1,296$.

2.6.1 Graph-theoretic problems

The natural action of $S(V)$ on V^2 gives rise to the NPC SUBGRAPH ISOMORPHISM problem for a pair of directed graphs $H_1 = \langle V, E_1 \rangle$, $H_2 = \langle V, E_2 \rangle$ over a finite set of vertices V:

Is H_1 isomorphic to a subgraph of H_2? Formally, is it true that $(\exists g \in S(V))\ E_1^g \subseteq E_2$?

Two special cases of this problem are GRAPH ISOMORPHISM when the two graphs must have the same number of edges, and GRAPH AUTOMORPHISM where we have $H_1 = H_2$ and we require a map in $S(V)\backslash 1$. Luks shows GRAPH ISOMORPHISM $\leq_P$ OPS [Luk82]. GRAPH ISOMORPHISM is not known to be NPC and we briefly encounter some of the issues surrounding this problem in §8.3.3.

2.7 Groups acting on boolean structures

The material of this section is synthesised from accounts in [Mar98, Har71]. We review some of the basic concepts of groups acting on boolean functions and boolean terms. Perhaps the most natural group acting on boolean functions $\mathcal{B}_n$ is the symmetric group S_n of order $n!$ acting by permuting the arguments of the function. Namely if $g \in S_n$ then define

$$F^g \stackrel{\text{def}}{=} \langle b_1, \ldots, b_n \rangle \mapsto F(b_{1^{g^{-1}}}, \ldots, b_{n^{g^{-1}}}).$$

It is easy to check that this gives an S_n action on $\mathcal{B}_n$ and thus a group of invertible transforms on the functions. The reason the inverse of g is applied to the indices is that this corresponds to the natural action of g on 2^n by

$$\langle b_1, \ldots, b_n \rangle^g \stackrel{\text{def}}{=} \langle b_{1^g}, \ldots, b_{n^g} \rangle$$

and thus on B_n viewed as $\mathcal{P}(2^n)$. I.e. for $F \subseteq 2^n$ we have $F^g = \{a^g \mid a \in F\}$ and so

$$F^g(b_{1^g}, \ldots, b_{n^g}) = F(b_1, \ldots, b_n).$$

Another natural group acting on $\mathcal{B}$ is the complementation group $S(2)^n$, the direct product of n copies of the symmetric group on $\{0, 1\}$. We will make this act by flipping the polarity of the arguments to the functions. Let $\delta = \langle \delta_1, \ldots, \delta_n \rangle \in S(2)^n$ and define

$$F^\delta = \langle b_1, \ldots, b_n \rangle \mapsto F(b_1^{\delta_1}, \ldots, b_n^{\delta_n}).$$

Again we have an action of the group on $\mathcal{B}_n$ giving a group of invertible transforms the functions. The groups $S(2)$ and S_n can be combined to induce an action of the wreath product $S(2) \wr S_n$ on $\mathcal{B}_n$. Following §2.5 we could realise this as a subgroup G_n of $S(2 \times \{1, \ldots, n\})$ where $S(2)^n$ corresponds to the base of the wreath product. The group has order $n!2^n$.

2.7.1 Groups acting on boolean terms

Proceeding similarly we start by defining two corresponding infinite groups over the set Lit of literals as follows:

$$\Delta \stackrel{\text{def}}{=} \langle (x\ \neg x) \mid x \in \Omega \rangle$$
$$S(\Omega)^{\neg} \stackrel{\text{def}}{=} S(\Omega)[\mathsf{Lit}] \text{ with } (\neg x)^g = \neg x^g.$$

Δ is the group generated by all independent 'flips' between x and $\neg x$ and $S(\Omega)^{\neg}$ is the symmetric group on Ω with a naturally induced parallel action on negative literals. It can be checked that the group $W \stackrel{\text{def}}{=} \Delta S(\Omega)^{\neg}$ is indeed a group and is isomorphic to the infinite group $S(2) \wr S(\Omega)$. Δ is the base of this wreath product.

We wish to define an action of W on the boolean terms Wff. Since Lit $\subset$ Wff then we can define the action in terms of a Wff-assignment on Ω (recall that Wff is a synonym for the B-algebra $\mathcal{T}_B(\Omega)$) as follows. View $g \in W$ as a Wff-assignment on Ω sending $x \in \Omega$ to the literal x^g. Then by the Freeness Theorem for Σ-algebras we have a unique map $g^{\#} : \mathsf{Wff} \to \mathsf{Wff}$ agreeing with g on Ω. We need a stronger notion of equivalence to be able to view $\#$ as a group action, putting $\phi^g \stackrel{\text{def}}{=} g^{\#}(\phi)$. This is because in the free structure Wff we do not have the double negation rule $\neg(\neg x) = x$ for $x \in \Omega$ and so substitutions $g^{\#}, h^{\#}$ applied in succession may not correspond to $(gh)^{\#}$. However if we have a congruence $=_E$ on terms for some set of equations containing the double negation rule, then we can define the action successfully on the E-congruence classes of Wff. In particular we have that the double negation rule is a consequence of the boolean algebra equations BA and that $=_{BA}$ defines standardly the same relation as $\equiv$. This boils down to the fact that in the context of $\models$ or $\equiv$ we can substitute $((\phi)^w)^v)$ for ϕ^{wv}

since they are equivalent under double negation and hence under logical equivalence, and therefore we have a group action of W on Wff with respect to logical equivalence.

It is important to see the group W arising in general terms as a set of invertible transforms on Wff modulo a suitable equivalence, but in fact we will nearly always operate with finite subgroups $W(\vec{x})$ for sequences $\vec{x} \subset \Omega$. With $|\vec{x}| = n$ and the finite group $W(\vec{x})$, we have a group isomorphic to G_n so we could in fact use either. However we prefer the links to the term algebra approach, set up in this section, within the body of the thesis: in reductions and group manipulations of the formulas in a complexity context it is the term structure that is of prevailing importance rather than the underlying function.

In reiteration of a general convention used in the thesis: the relevant propositional letters in theorems and statements concerning formulas and groups are introduced by expressions $\phi(\vec{x}), \psi(\vec{x}), \ldots$ which denote that the variables occurring in $\phi, \psi, \ldots$ are contained (perhaps strictly) in the finite sequence $\vec{x}$; then we induce finite structures $W(\vec{x}), \mathsf{Lit}(\vec{x}), \Delta(\vec{x}), \ldots$ from the infinite ones $W, \mathsf{Lit}, \Delta, \ldots$.

In Appendix A we give an account of some properties of groups acting on binary decision diagrams. This is not a central issue but accompanies some work on BDDs elsewhere in the thesis.

2.8 W home page

In this section we collect together some commonly used facts and identities concerning our most prevalent group, W.

Notation Since the image of $x \in \Omega$ under $g \in W$ determines the image of $\neg x$, we adopt a shorthand notation from [BdlT96a] for describing elements of W in cycle notation. For instance the permutation $(x\ \neg y\ z)(\neg x\ y\ \neg z)$ would be written in the

annotated form $(x\ \neg y\ z)^{\neg}$. Other example elements of $W(x,y,z)$ are

$$(x\ y\ z)(\neg x\ \neg y\ \neg z) \text{ or just } (x\ y\ z)^{\neg}$$
$$(x\ \neg y\ \neg z\ \neg x\ y\ z)$$
$$(x\ \neg x)(y\ \neg y)(z\ \neg z).$$

Action of W on assignments We define the action of W on the assignments Asg by

$$a^g \stackrel{\text{def}}{=} x \mapsto x^{g^{-1}}(a)$$

(note we cannot put $a(x^{g^{-1}})$ since $x^{g^{-1}}$ may be a literal requiring evaluation). Then by checking that the two B-homomorphisms $(a^g)^{\#}$ and $a^{\#} \circ (g^{-1})^{\#}$ agree on Ω we have for all $\phi \in \mathsf{Wff}$

$$\phi(a^g) = \phi^{g^{-1}}(a).$$

Illustrating the connection between models of wff $\phi(\vec{x})$ and the action of $W(\vec{x})$ on assignments we have

$$\begin{aligned}
\mathsf{Mod}(\phi^g) &= \{a \in \mathsf{Asg}(\vec{x}) \mid \phi^g(a) = 1\} \\
&= \{a \in \mathsf{Asg}(\vec{x}) \mid \phi(a^{g^{-1}}) = 1\} \\
&= \{a^g \in \mathsf{Asg}(\vec{x}) \mid \phi(a) = 1\} \\
&= \{a \in \mathsf{Asg}(\vec{x}) \mid \phi(a) = 1\}^g \\
&= \mathsf{Mod}(\phi)^g.
\end{aligned}$$

On defined connectives We check that W behaves properly with respect to the defined connectives $\rightarrow, \leftrightarrow, +$. It is straightforward that for $\phi, \psi \in \mathsf{Wff}$ and $g \in W$

$$(\phi \rightarrow \psi)^g \equiv \phi^g \rightarrow \psi^g$$

and so on for the other connectives.

Validity and group action An identity we use frequently is the fact that a valid (or unsatisfiable) formula remains valid (or unsatisfiable) under the action of any map $g \in W$.

Lemma 2.8.1 $\models \phi \Leftrightarrow \models \phi^g$

The result follows easily from considering that the set of assignments Asg is closed under W-action.

Centre of W The *centre* of a group G is the set

$$\text{Centre}(G) \stackrel{\text{def}}{=} \{g \in G \mid (\forall h \in G) gh = hg\}.$$

The centre of W is the group consisting of just the identity and the map

$$f = (x_1\ \neg x_1)(x_2\ \neg x_2)\cdots$$

which is the map which flips polarity of all variables.

2.8.1 The Δ-Asg correspondence

We shall refer occasionally to a correspondence between the group Δ and the set Asg of **2**-assignments on Ω. The set Asg can in fact be viewed as a group isomorphic to Δ if we think of it as the set of functions from Ω into the *group* $S(2)$ rather than the set 2. Then the composition defined by $(ab)(x) \stackrel{\text{def}}{=} a(x)b(x)$ gives rise to a group which is isomorphic to Δ. Recall that Δ arose as the base of the wreath product W. The set of assignments Asg, viewed as a group, is an example of how the base arises in the abstract wreath product (see for example [DM96, §2.6]). It is this correspondence that facilitates the transference of some logical problems into more general group theoretical ones in Chapters 5–7. However as we shall point out frequently the structure of groups Δ and W are not crucial to how this correspondence helps in the proofs.

2.9 Symmetry

Having fixed the details of the actions for the groups and the structures we are interested in, we revert to the familiar concept of the symmetry, or invariance, group of a boolean function or term. Note that although the principle of 'invariance' is essentially the same as that of 'automorphism' (which we have already met in the context of graphs) the former concept emphasises that we are dealing now with individual elements of a structure, and not whole structures.

Symmetry of functions For functions $F \in \mathcal{B}_n$ define *the symmetry group of* F *in the group* $H \leq G_n$ by

$$\mathsf{Stab}(F, H) = \{g \in H \mid F^g = F\}.$$

It is straightforward to confirm that this set is indeed closed under compositions and inverses and is therefore a subgroup of H.

Symmetry of wffs Define the *symmetry group* of $\phi(\vec{x})$ in the group $G \leq W(\vec{x})$ as:

$$\Sigma(\phi, G) \stackrel{\text{def}}{=} \{g \in G \mid \phi^g =_{BA} \phi\}$$

that is, the subset (which again is straightforwardly a group) of G which fixes the BA-congruence class of ϕ. Since we have standardly that $=_{BA}$ and $\equiv$ define the same relation, then the notions of symmetry using either relation will be used interchangeably. When G is the group $W(\vec{x})$ we abbreviate by

$$\Sigma(\phi) \stackrel{\text{def}}{=} \Sigma(\phi, W(\vec{x})).$$

This corresponds to the definition of *semantic* symmetry group of a formula given by [BdlT96a]. We will encounter the notion of *syntactic* symmetries of formulas in §3.2.

2.9.1 Symmetry properties

This section contains a few routine lemmas about symmetry that will be used in later chapters.

Symmetry and $\rightarrow$ versus $\leftrightarrow$ We can relax the double-implication notion of symmetry to one of implication.

Lemma 2.9.1 For any wff $\phi(\vec{x})$ and $g \in W(\vec{x})$ we have $\models \phi^g \rightarrow \phi \;\Leftrightarrow\; \models \phi^g \leftrightarrow \phi$.

Symmetry and generators Symmetry under generators is sufficient for the whole group.

Lemma 2.9.2 Let $\phi(\vec{x})$ be a wff and $G \leq W(\vec{x})$ be generated by $g_1, \ldots, g_k$. If $\models \phi^{g_k} \leftrightarrow \phi$ for each generator g_k of G then $\models \phi^g \leftrightarrow \phi$ holds for all $g \in G$.

We conclude the section on symmetry with two issues of importance that will be referred to occasionally but which are not central to the thesis.

2.9.2 Representability

A fascinating problem that has been addressed recently in the literature, but which does not apply directly to the work of this thesis, is the question of *representability* of groups $G \leq S_n$ as the symmetry group of a boolean function over n variables. It turns out that not all groups can be so represented. See [CK91] and a subsequent paper by Kisielewicz [Kis98]. The simplest example is perhaps the cyclic group C_3. Any boolean function on three variables invariant under this group is also invariant under the symmetric group S_3, a fact that is invoked in the graph example of §3.5.1. Less trivial examples are quite easy to find. There is a transitive group of degree 12 such that any element of $\mathcal{B}_{12}$ invariant under this group of symmetries must also be invariant under a group 72 times larger[1]. We will refer to the issue of representability occasionally in the context of other methods for inferring more symmetries of formulas when some group is already known.

2.9.3 Number of functions with symmetry

Given that symmetry of boolean functions, and their representations as boolean terms, is one of the main themes of this thesis, the question arises as to how many functions there are with a nontrivial symmetry group. We could phrase this as follows: given a random boolean function $F \in \mathcal{B}_n$ what is the probability that F is fixed by some nontrivial element of either S_n or G_n? The answer is that the probability is extremely small. Clote and Kranakis showed for S_n that the probability tends to $n!2^{-2^{n-2}}$ as n tends to infinity [CK91]. Clausen later showed that the result for 'linear' symmetries (the n-dimensional general linear group $\mathrm{GL}(n, 2)$ over a field of two elements is the set of all invertible $n \times n$ matrices over 2, and is larger than G_n) was asymptotically the same, using a somewhat simpler argument [Cla92].

[1] The group $[(L(6) : 2)^2]2$ available as `TransitiveGroup(12,288)` in GAP.

The sparseness results are to some extent ameliorated if we consider the types of formulas that arise from natural problems which often contain a large measure of symmetry. Chapter 4 looks at some of these problems and how researchers have exploited the symmetries in them.

In particular, a formula that is valid or false is logically invariant under all maps (cf. Lemma 2.8.1). Such a formula is not however trivially symmetrical in the same sense that the underlying function is. We see in Chapter 5 that the intractability of determining whether a formula is logically invariant under just one map is directly linked to the intractability of the problem UNSAT.

2.10 Summary

We have collected core background material for the rest of the thesis. The index to the thesis contains pointers to most of the notation. The next chapter examines a number of transformation techniques that will be applied regularly in subsequent chapters.

Chapter 3

Tools and Constructions

3.1 Introduction

We collect together and review in this chapter five transformation techniques for manipulating logical and group-theoretic problems. The techniques are of general applicability throughout the thesis. We motivate each construction and give examples. The chapter may be scanned briefly and referred back to as necessary. The following chapter plan contains a summarised account of the chapter contents which may be sufficient in this respect.

3.1.1 Plan of chapter

We outline a construction in §3.2 derived from a Galois connection between boolean functions $\mathcal{B}_n$ and the clause systems over n variables where each clause has exactly $k \leq n$ literals from distinct variables. We adopt an approach with the aim of not making the Galois connection notions too formal but we briefly relate our attempt at formalisation to Martin's version [Mar98] in a summarising section, as well as saying what a Galois connection is. This construction constitutes a problem transformation technique in that it provides an operator on wffs which has a high computational cost in most cases but which subsequently simplifies many problems concerning group action, equivalence and containment. Despite the high cost we show for problems in Chapters 6 and 7 cases where we get a better than worst case result for some classes of formula

which might not be expected to exhibit this improved behaviour.

In §3.3 and §3.4 we examine two constructions for manipulating formulas in the context of group action. The first of these is a construction for 'gluing' some formulas together with the underlying consideration being the requirement to exhibit problem *transformations*. Problem transformations require the target problem to be a single instance of the problem. For example the property that either of ϕ, ψ are valid can be expressed as the property that a single formula is valid where that single formula is obtained by gluing together versions of the original two. The second of these constructions, what we call the D-transformation, corresponds to a routine technique in representing groups and enables us to reason about the group $W(\vec{x})$ as a subgroup of a larger symmetric group. By applying the same principle to formulas, we can transfer properties combining formulas with group action to new formulas and groups where the new groups are subgroups of a symmetric group. Thus we can show in many cases that there is nothing special about problems stated using subgroups of $W(\vec{x})$ compared with $S(\vec{x})$, though certain characteristics of $W(\vec{x})$ subgroups, discussed in §2.8.1, allow some of our later proofs to be simplified.

In order to show that problems are computationally equivalent to, or harder than, problems in a different domain, one has to show how to encode one problem in terms of another. We review several constructions in §3.5 that have been used to encode graphs, sets and wffs in terms of one another in various directions. We mention the results that have been obtained previously using these constructions.

A well-known problem transformation technique of great practical significance is the construction of stabiliser chain representations for finite permutation groups. Once computed, the representation permits a number of seemingly difficult problems to be solved with great efficiency. We review some of the relevant details and applications in §3.6.

3.2 Closure operators for k-CNF and k-DNF

Given some $k \leq n$ we may associate with any boolean function $F \in \mathcal{B}_n$ a *maximal* k-CNF formula which consists of the conjunction of those clauses entailed by F which are of length k and whose literals are from distinct variables. In the case where F is representable as a k-CNF formula ϕ, then the maximal k-CNF formula ψ associated with F is equivalent to ϕ and has some useful properties. For example any other k-CNF formula representing F is 'contained' in ψ, that is, its clauses all occur in ψ. Furthermore operations such as logical implication and equivalence can be replaced with the operations of inclusion and equality for pairs of maximal formulas viewed as sets of clauses. There is also a natural way of combining group action with these 'closed' formulas which reduces all issues of semantic symmetries to syntactic ones. Below we formalise this correspondence and we will use it for complexity proofs of Chapters 6–7. The observation of these properties and their formalisation as a Galois connection is due to [Mar98], and we briefly review this idea in a summarising section. Our definitions here are with practicality as the primary consideration.

The above discussion reveals that it is more convenient in many cases to view k-CNF formulas as combinatorial objects consisting of sets of sets of literals. In the following section we introduce notation to handle this and to convert between the two representations. We have also a corresponding set of properties for the dual structures, the k-DNF formulas and we fix a notation below for dealing with duality in a systematic way.

We introduce some notation to avoid repeated dual statements about CNFs and DNFs. Let variable τ range over a type $\{C, D\}$ and put

$$\overline{C} \stackrel{\text{def}}{=} D \qquad \overline{D} \stackrel{\text{def}}{=} C.$$

We may now talk about properties of k-τNF formulas and so on, whose duals are k-$\overline{\tau}$NF formulas. Many of the properties below can thus be stated in respect of k-τNF formulas with τ equal to either C or D.

3.2.1 Definitions

The literals Lit are as defined in §2.3.2. As sets, the *clauses* Cl_k containing exactly k literals from distinct variables are defined by

$$\mathsf{Cl}_k \stackrel{\text{def}}{=} \{s \subset \mathsf{Lit} \mid |s| = k, (\forall x \in \Omega)\{x, \neg x\} \not\subseteq s\}.$$

As usual we may put $\mathsf{Cl}_k(\vec{x})$ to induce the finite set of k-clauses over variables $\vec{x}$.

The group $W(\vec{x}) < S(\mathsf{Lit}(\vec{x}))$ acts on $\mathsf{Cl}_k(\vec{x})$ by natural extension from the action on $\mathsf{Lit}(\vec{x})$ cf. §2.5. Fix $k \geq 1$ and let $\alpha, \beta \subseteq \mathsf{Cl}_k(\vec{x})$ in the following. We use the τ notation to define maps from $\mathsf{Cl}_k(\vec{x})$ to wffs as follows:

$$\begin{aligned} \alpha^C &\stackrel{\text{def}}{=} \bigwedge\{\bigvee s \mid s \in \alpha\} \\ \alpha^D &\stackrel{\text{def}}{=} \bigvee\{\bigwedge s \mid s \in \alpha\}. \end{aligned}$$

These maps $\tau \in \{C, D\}$ send sets of sets of literals to k-τNF formulas over variables $\vec{x}$. Note that $\bigwedge\{\} \stackrel{\text{def}}{=} 1$ and $\bigvee\{\} \stackrel{\text{def}}{=} 0$.

Next we define a map from wffs to subsets of $\mathsf{Cl}_k(\vec{x})$ as follows:

$$M_{k,\tau,\vec{x}}(\phi) \stackrel{\text{def}}{=} \begin{cases} \{s \in \mathsf{Cl}_k(\vec{x}) \mid \phi \models \bigvee s\} & \text{if } \tau = C \\ \{s \in \mathsf{Cl}_k(\vec{x}) \mid \bigwedge s \models \phi\} & \text{if } \tau = D \end{cases}$$

giving us the $J_{k,\tau,\vec{x}}$ operator on wffs defined by

$$J_{k,\tau,\vec{x}}(\phi) \stackrel{\text{def}}{=} M_{k,\tau,\vec{x}}(\phi)^\tau.$$

We say $\phi(\vec{x})$ is k-τNF *representable* if $J_{k,\tau,\vec{x}}(\phi) \equiv \phi$. If $\phi(\vec{x})$ is k-τNF representable then $J_{k,\tau,\vec{x}}(\phi)$ is the maximal k-τNF representation of ϕ discussed in the introduction to this section. $M_{k,\tau,\vec{x}}(\phi)$, being a system of clauses, gives us an intermediate representation to which we can apply set operations.

Note that often we will be applying $M_{k,\tau,\vec{x}}$ and $J_{k,\tau,\vec{x}}$ when $\mathsf{Vars}(\phi)$ might be a strict subset of $\vec{x}$: this is why $\vec{x}$ is included in the notation.

3.2.2 Properties

Lemma 3.2.1 If wff $\phi(\vec{x})$ is in k-τNF then $J_{k,\tau,\vec{x}}(\phi) \equiv \phi$.

That is, any k-τNF formula is clearly k-τNF representable. The first three statements of the following lemma are for $\tau = C$. The statements assert: i) the contraction property of the closure operator; ii) the order-reversing property associated with a Galois connection (see §3.2.3); iii) a strengthened order-reversing property in the case that ψ is k-CNF representable. The final statement asserts the dual properties for $\tau = D$.

Lemma 3.2.2 The following properties hold:

i). For any wff $\phi(\vec{x})$ then $\phi \models J_{k,C,\vec{x}}(\phi)$.

ii). For $\alpha, \beta \subseteq \mathsf{Cl}_k(\vec{x})$ and wffs $\phi(\vec{x}), \psi(\vec{x})$:

$$\alpha \subseteq \beta \;\Rightarrow\; \beta^C \models \alpha^C$$
$$\phi \models \psi \;\Rightarrow\; M_{k,C,\vec{x}}(\psi) \subseteq M_{k,C,\vec{x}}(\phi).$$

iii). For wffs $\phi(\vec{x}), \psi(\vec{x})$ if ψ is k-CNF representable then

$$\phi \models \psi \;\Leftrightarrow\; M_{k,C,\vec{x}}(\psi) \subseteq M_{k,C,\vec{x}}(\phi).$$

iv). Statements i)–iii) also hold when C is replaced by D and $\models$ is replaced by $=\!|$.

Proof Statements i) and ii) follow easily from the definitions of the previous section. For iii) the $\Rightarrow$ direction follows by applying C to both closures, followed by an application of the order-reversing rule ii), and then using i) and the assumption that $\psi \equiv J_{k,C,\vec{x}}(\psi)$. The dual properties follow by dual proofs.□

The following lemma relates group action on formulas and clause systems. Statements i) and ii) are straightforward properties. Statements iii) and iv) relate the semantic or logical symmetries of formula ϕ in group G to the setwise stabiliser of the closure.

Lemma 3.2.3 Let $\phi(\vec{x})$ be a wff and $g \in G \leq W(\vec{x})$. The following properties hold:

i). $M_{k,\tau,\vec{x}}(\phi^g) = M_{k,\tau,\vec{x}}(\phi)^g$.

ii). If $J_{k,\tau,\vec{x}}(\phi) \equiv \phi$ then $J_{k,\tau,\vec{x}}(\phi^g) \equiv \phi^g$.

iii). $\Sigma(\phi, G) \leq \mathsf{Stab}(M_{k,\tau,\vec{x}}(\phi), G)$.

iv). If ϕ is k-τNF representable then $\Sigma(\phi, G) = \mathsf{Stab}(M_{k,\tau,\vec{x}}(\phi), G)$.

Proof Statements i) and ii) are direct. For iii) let $g \in \Sigma(\phi, G)$: then $\phi \equiv \phi^g$ and so $M_{k,\tau,\vec{x}}(\phi) = M_{k,\tau,\vec{x}}(\phi^g)$ by order-reversing applied both ways. Applying i) then gives that g is an element of the stabiliser of the closure of ϕ. For iv) we can sandwich the equation between these inequalities:

$$\begin{array}{rcll} \Sigma(\phi, G) & \leq & \mathsf{Stab}(M_{k,\tau,\vec{x}}(\phi), G) & \text{by iii)} \\ & \leq & \Sigma(J_{k,\tau,\vec{x}}(\phi), G) & \text{syntactic} \leq \text{semantic} \\ & = & \Sigma(\phi, G) & \text{by representability assumption.} \end{array}$$

□.

Example 3.2.4 Let $\vec{x} = x, y, z$ and put

$$\begin{aligned} \phi &\stackrel{\text{def}}{=} x \wedge y \wedge z \\ \psi &\stackrel{\text{def}}{=} (x \vee \neg y) \wedge (y \vee \neg z) \wedge (z \vee \neg x). \end{aligned}$$

We illustrate how $\phi \models \psi$ can be determined by applying the $M_{k,\tau,\vec{x}}$ operator, reducing the problem to testing $\subseteq$; and we illustrate how $\Sigma(\phi)$ can also be computed by reduction to computing a setwise stabliser of $M_{k,\tau,\vec{x}}(\phi)$. Put

$$\begin{aligned} \alpha &\stackrel{\text{def}}{=} M_{2,C,\vec{x}}(\phi) = \{\{x, \neg y\}, \{y, \neg z\}, \{z, \neg x\}, \{\neg x, y\}, \{\neg y, z\}, \\ &\qquad\qquad \{\neg z, x\}, \{x, y\}, \{y, z\}, \{x, z\}\} \\ \beta &\stackrel{\text{def}}{=} M_{2,C,\vec{x}}(\psi) = \{\{x, \neg y\}, \{y, \neg z\}, \{z, \neg x\}, \{\neg x, y\}, \{\neg y, z\}, \{\neg z, x\}\}. \end{aligned}$$

Then converting α back to 2-CNF we have $\alpha^C = J_{2,C,\vec{x}}(\phi) \equiv \phi$. So ϕ is 2-CNF representable. Therefore since $\beta \subseteq \alpha$ we have by Lemma 3.2.2 that $\phi \models \psi$. Similarly we can compute 3-DNF representations

$$\begin{aligned} \alpha &\stackrel{\text{def}}{=} M_{3,D,\vec{x}}(\phi) = \{\{x, y, z\}\} \\ \beta &\stackrel{\text{def}}{=} M_{3,D,\vec{x}}(\psi) = \{\{x, y, z\}, \{\neg x, \neg y, \neg z\}\}. \end{aligned}$$

Converting α to 3-DNF then we have $\alpha^D = J_{3,D,\vec{x}}(\phi) \equiv \phi$ so ϕ is 3-DNF representable. Therefore since $\alpha \subseteq \beta$ we have by Lemma 3.2.2 that $\psi \nvDash \phi$, or $\psi \models \phi$ as before.

But ϕ is not 2-DNF representable: $M_{2,D,\vec{x}}(\phi) = \{\}$ and so $J_{2,D,\vec{x}}(\phi) = \{\}^D = 0 \not\equiv \phi$.

Inspection reveals the following concerning symmetries:

$$\mathsf{Stab}(M_{2,C,\vec{x}}(\psi), W(\vec{x})) = \langle S(\vec{x}), f(\vec{x})\rangle$$

where $f(\vec{x})$ is the nontrivial permutation in the centre of $W(\vec{x})$ (see §2.8). Since ψ is 2-CNF representable, as noted above, then by Lemma 3.2.3

$$\Sigma(\psi) = \langle S(\vec{x}), f(\vec{x})\rangle.$$

That is, the semantic symmetry group of ψ (in $W(\vec{x})$) is the same as the setwise stabiliser of $M_{2,C,\vec{x}}(\psi)$ with $W(\vec{x})$ acting on clauses in the natural way.□

3.2.3 Note on Galois connections

The relationship between Wff and $\mathsf{Cl}_k(\vec{x})$ established by the operators $M_{k,\tau,\vec{x}}$ and τ almost define the Galois connection construction given by Martin in [Mar98]. Our formalisation is not intended to represent the construction given there, but is derived wholly from it. Briefly, a pair $(\lhd, \rhd)$ of maps $\lhd : Q \to P$, $\rhd : P \to Q$ between two partial orders $(P, \leq)$, $(Q, \leq)$ is called a *Galois connection* if it obeys

$$p \leq q^{\lhd} \Leftrightarrow q \leq p^{\rhd}$$

from which property versions of Lemma 3.2.2 can be derived (see for example [PD94]). If we take $\mathsf{Wff}_{/BA}$ instead of Wff then we can discard the preorder $\models$ in favour of the induced partial ordering $\leq$

$$[\phi]_{BA} \leq [\psi]_{BA} \stackrel{\text{def}}{=} [\phi \vee \psi]_{BA} = [\psi]_{BA}$$

We have chosen to retain $\models$ as the ordering on wffs because it represents the policy that in the body of this thesis we are concerned entirely with term representations of boolean functions and to revert back to dealing with the functions (i.e. the formulas modulo $=_{BA}$) would add an extra layer of interpretation to the usage we make of the

properties of $M_{k,\tau,\vec{x}}$ in the following chapters. Although technically our construction $(M_{k,\tau,\vec{x}}, \tau)$ does not give a Galois connection since $\models$ is not a partial order, it seems we can get most of the useful standard properties. Indeed the structure suggested by the Galois connection is very helpful in providing the most suitable base of lemmas to reason with in following chapters.

3.3 Combining formulas together

A reduction technique used several times in the following chapters is the combining of statements of validity of several formulas into a statement of validity of a single expression. For example, we may wish to combine $\models \phi_1 \leftrightarrow \psi_1$ and $\models \phi_2 \leftrightarrow \psi_2$ into a single equivalent statement of the form $\models \langle \phi_1, \phi_2 \rangle \leftrightarrow \langle \psi_1, \psi_2 \rangle$. We formalise the constructions and emphasize any conditions upon which the equivalences hold.

To capture this idea of a product we must have some machinery to standardise apart a set of formulas, that is, reformulate them using disjoint sets of variables. We do this using permutations to map variables of a formula to a new set of variables. These renaming permutations can then be combined with group action on the formulas so that a number of disjoint group-theoretic questions about the formulas can be glued together into a single question. The point of this is to be able to exhibit problem transformations where typically a reduction generates several statements which then have to be glued together into a single instance of the target problem.

3.3.1 Preliminaries

Let $\vec{x}, \vec{y}$ be disjoint with $|\vec{x}| = |\vec{y}|$. Let $h \in S(\vec{x}, \vec{y})$ be the transposition such that $\vec{x}^h = \vec{y}$. Since $\vec{x}, \vec{y}$ are disjoint and of equal length, then the assignments $\mathsf{Asg}(\vec{x}, \vec{y})$ may be identified with the product $\mathsf{Asg}(\vec{x}) \times \mathsf{Asg}(\vec{x})$ and thus we write $\langle a, b \rangle$ for $a, b \in \mathsf{Asg}(\vec{x})$ to range over the elements of $\mathsf{Asg}(\vec{x}, \vec{y})$.

3.3.2 Properties

Lemma 3.3.1 For wffs $\phi(\vec{x}), \psi(\vec{x})$ then

$$\models \phi \vee (\psi^h) \;\Leftrightarrow\; \models \phi \text{ or } \models \psi.$$

Proof Say neither ϕ nor ψ are valid. Then there are assignments $a, b \in \mathsf{Asg}(\vec{x})$ such that $\phi(a) = 0$ and $\psi(a) = 0$. Hence there is an assignment $\langle a, b\rangle$ such that $(\phi \vee \psi^h)(\langle a, b\rangle) = 0$ and therefore $\phi \vee \psi^h$ is not valid. Conversely, if either ϕ or ψ are valid, then $\phi \vee \psi^h$ is clearly valid.□

In the conjunctive case we put $\langle \phi, \psi\rangle \stackrel{\text{def}}{=} \phi \wedge (\psi^h)$. In the following lemma the satisfiability criterion is necessary to obtain the intuitive correspondence[1].

Lemma 3.3.2 Let wffs $\phi_1(\vec{x}), \phi_2(\vec{x}), \psi_1(\vec{x}), \psi_2(\vec{x})$ be satisfiable. Then

$$\models \langle \phi_1, \phi_2\rangle \leftrightarrow \langle \psi_1, \psi_2\rangle \;\Leftrightarrow\; \models \phi_1 \leftrightarrow \psi_1 \;\&\; \models \phi_2 \leftrightarrow \psi_2.$$

Proof If RHS then for all $a, b \in \mathsf{Asg}(\vec{x})$ we have $\phi_1(a) = \psi_1(a)$ and $\phi_2(b) = \psi_2(b)$. Therefore for all $\langle a, b\rangle \in \mathsf{Asg}(\vec{x}, \vec{y})$ we have $\langle \phi_1, \phi_2\rangle(\langle a, b\rangle) = \langle \psi_1, \psi_2\rangle(\langle a, b\rangle)$. Therefore LHS is valid. Conversely, say $\not\models \phi_1 \leftrightarrow \psi_1$. Then there is an assignment $a \in \mathsf{Asg}(\vec{x})$ such that $\phi_1(a) \neq \psi_1(a)$. Say $\phi_1(a) = 1$ and $\psi_1(a) = 0$. Then, since by assumption ϕ_2 is satisfiable, there exists $b \in \mathsf{Asg}(\vec{x})$ such that $\phi_2(b) = 1$. Then $\langle \phi_1, \phi_2\rangle(\langle a, b\rangle) = 1$ and $\langle \psi_1, \psi_2\rangle(\langle a, b\rangle) = 0$. So LHS is not valid. The proof is completed by backtracking to consider $\phi_1(a) = 0$ and $\psi_1(a) = 1$ and then the case where $\not\models \phi_2 \leftrightarrow \psi_2$, which cases admit similar proofs.□

Lemma 3.3.2 extends to the general case where we use transpositions h_i from $\vec{x_0}$ to $\vec{x_i}$ for disjoint sequences $\vec{x_k}$ of the same cardinality. We put $h_0 = \epsilon$ and then write

$$\langle \phi_1, \ldots, \phi_i\rangle \;\stackrel{\text{def}}{=}\; \phi_1^{h_0} \wedge \ldots \wedge \phi_i^{h_i}.$$

---

[1] Say ϕ_1, ψ_2 are unsatisfiable. Then LHS but ψ_1 may be satisfiable and so RHS is not true. This was not initially observed and led to some incorrect proofs.

Lemma 3.3.3 Let wffs $\phi_1(\vec{x}), \ldots, \phi_i(\vec{x}), \psi_1(\vec{x}), \ldots, \psi_i(\vec{x})$ be satisfiable . Then

$$\models \langle \phi_1, \ldots, \phi_i \rangle \leftrightarrow \langle \psi_1, \ldots, \psi_i \rangle \;\Leftrightarrow\; \models \phi_1 \leftrightarrow \psi_1 \;\&\; \ldots \;\&\; \models \phi_i \leftrightarrow \psi_i.$$

The proof is by routine extension of the proof of Lemma 3.3.2. Note that if $G \leq W(\vec{x})$ then the group generated by G and $h_0, \ldots, h_i$ is a permutation representation of the wreath product $G \wr S_i$ where the base is $(h_0{}^{-1}Gh_0) \ldots (h_i{}^{-1}Gh_i)$, the set product of i conjugates of G acting on disjoint points.

We show how these devices can be used in two examples.

Example 3.3.4 Let $\phi(\vec{x})$ be a wff and $G \leq W(\vec{x})$. Find an element $\gamma \in \langle G, h_1, h_2, h_3, h_4 \rangle$ such that $\langle \phi_1, \phi_2, \phi_3, \phi_4 \rangle^\gamma \mapsto \langle \phi_3, \phi_4, \phi_1, \phi_2 \rangle$: the permutation (1 3)(2 4) of the components can be represented as the product (1 3)(1 2)(1 4)(1 2) of transpositions and hence the required γ is $h_3h_2h_4h_2$.□

Example 3.3.5 Let $g_1, g_2, g_3 \in G \leq W(\vec{x})$. For wff $\phi(\vec{x})$, find an element $\gamma \in \langle G, h_1, h_2, h_3 \rangle$ such that $\langle \phi_1, \phi_2, \phi_3 \rangle^\gamma \mapsto \langle \phi_1^{g_1}, \phi_2^{g_2}, \phi_3^{g_3} \rangle$: the required permutation is the product of the conjugates of g_i under h_i, i.e. $(h_1^{-1}g_1h_1)(h_2^{-1}g_2h_2)(h_3^{-1}g_3h_3)$ which we can write $\langle g_1, g_2, g_3 \rangle$.□

By a composition of maps, we can simultaneously apply a vector of group elements to the product and switch the components around by any permutation.

3.4 The D-transformation

We address two problems associated with subgroups of $W(\vec{x})$ containing non-positive elements, that is maps which 'flip' the polarity of literals as well as permuting the variables. The first is of a general nature and asks the question whether the problems of the following chapters associated with subgroups of $W(\vec{x})$ are harder than for subgroups of $S(\vec{x})^\neg < W(\vec{x})$, which we refer to here as the 'positive' subgroups of $W(\vec{x})$.

The second problem is of a more pragmatic nature and concerns the construction of small satisfiable non-tautologous formulas fixed by some given group of symmetries.

This is trivial for positive groups since the formula $\bigwedge\vec{x} = x_1 \wedge \ldots \wedge x_n$ is of course invariant under any subgroup of $S(\vec{x})^\neg$. In general it is not so easy to find a small wff $\phi(\vec{x})$ fixed by a given non-positive group $G \leq W(\vec{x})$, and for some it is impossible: for example the group $\Delta(\vec{x})$ under which only valid or invalid formulas are fixed. We describe a transformation on both formulas and groups which enables questions posed about non-positive groups to be reformulated equivalently in terms of positive ones, and which thus facilitates certain reductions requiring new (satisfiable) symmetrical formulas.

3.4.1 Definitions

First we define D on wffs. Let $\vec{x}$ and $\vec{x}'$ be disjoint sequences of variables with $|\vec{x}| = |\vec{x}'| = n$. For wff $\phi(\vec{x})$ we define $D(\phi) \stackrel{\text{def}}{=} \phi \wedge \zeta$ where

$$\zeta \stackrel{\text{def}}{=} (x_1 \leftrightarrow \neg x_1') \wedge \ldots \wedge (x_n \leftrightarrow \neg x_n').$$

The effect of D is to redefine the literals $x_i, \neg x_i$ in terms of two variables so that x_i corresponds to $x_i \wedge \neg x_i'$ and $\neg x_i$ corresponds to $\neg x_i \wedge x_i'$. The models of $D(\phi)$ are in 1-1 correspondence with the models v of ϕ and may be viewed as $\langle a, a' \rangle$ where $a'(x_i') \stackrel{\text{def}}{=} (\neg x_i)(a)$ for each $a \in \mathsf{Asg}(\vec{x})$ such that $\phi(a) = 1$.

Now we define D on groups. We saw in §2.7 that $W(\vec{x}) = \Delta(\vec{x})S(\vec{x})^\neg$ so that any element of $W(\vec{x})$ may be written δs with $\delta \in \Delta(\vec{x})$ and $s \in S(\vec{x})^\neg$. We define the map D on group elements, and then groups, in terms of a homomorphism of $W(\vec{x})$ into $S(\vec{x}, \vec{x}')$. That is we take $W(\vec{x})$ into a subgroup of the symmetric group on twice the number of points. Recall that $\Delta(\vec{x})$ is generated by $(x_i\ \neg x_i)$ for each $1 \leq i \leq n$ and that the symmetric group $S(\vec{x})^\neg$ is generated by $(x_1\ x_i)^\neg$ for each $2 \leq i \leq n$. Define

$$\begin{aligned} D((x_i\ \neg x_i)) &\stackrel{\text{def}}{=} (x_i\ x_i') \\ D((x_1\ x_i)^\neg) &\stackrel{\text{def}}{=} (x_1\ x_i)(x_1'\ x_i') \end{aligned}$$

Thus D transforms a variable flip into a transposition between two variables and turns a transposition on $\vec{x}$ into a matching pair of transpositions on $\vec{x}, \vec{x'}$. It is straightforward to check that $D(\delta s) \stackrel{\text{def}}{=} D(\delta)D(s)$ gives a homomorphism $W(\vec{x}) \to S(\vec{x}, \vec{x}')$. Therefore

we have that if $g_1, \ldots, g_k$ are generators for $G \leq W^X$, then the set $D(G) \stackrel{\text{def}}{=} \{D(g) \mid g \in G\}$ is the same as the group generated as $\langle D(g_1), \ldots, D(g_k)\rangle$. The following lemmas establishes some useful properties for D.

3.4.2 Properties

Lemma 3.4.1 For wffs $\phi(\vec{x}), \psi(\vec{x})$ we have:

$$\models \phi \to \psi \quad \Leftrightarrow \quad \models D(\phi) \to D(\psi) \tag{3.1}$$

$$\models \phi \leftrightarrow \psi \quad \Leftrightarrow \quad \models D(\phi) \leftrightarrow D(\psi). \tag{3.2}$$

Proof First we show that for any $\theta(\vec{x})$ that

$$\models \theta \quad \Leftrightarrow \models \zeta \to \theta.$$

$\Rightarrow$ is clear. Conversely, if $\not\models \theta$ then there is $a \in \mathsf{Asg}(\vec{x})$ such that $\theta(a) = 0$. Hence for θ viewed as $\theta(\vec{x}, \vec{x}')$ then $\theta(\langle a, a'\rangle) = 0$. However, $\zeta(\langle a, a'\rangle) = 1$ for all $a \in \mathsf{Asg}(\vec{x})$ and so $\not\models \zeta \to \theta$. Now to show (3.1)

$$\begin{aligned} \models D(\phi) \to D(\psi) \quad &\Leftrightarrow \quad \models \phi \wedge \zeta \to \psi \wedge \zeta \\ &\Leftrightarrow \quad \models \zeta \to (\phi \to \psi) \\ &\Leftrightarrow \quad \models \phi \to \psi \end{aligned}$$

with the last step being a consequence of the first result for θ. The second case (3.2) for $\leftrightarrow$ is analogous. □

Note that similar results do not hold for connectives $\vee, \wedge$ in place of $\to, \leftrightarrow$. Combining the two transformations we have:

Lemma 3.4.2 For wff $\phi(\vec{x})$ and $g \in W(\vec{x})$ then $D(\phi^g) \equiv D(\phi)^{D(g)}$.

Proof This is straightforward putting $g = \delta s$ for $\delta \in \Delta$ and $s \in S(\vec{x})^\neg$ with the gist of the argument being

$$D(\phi^{\delta s}) \equiv D((\phi^\delta)^s) \equiv D(\phi^\delta)^{D(s)} \equiv D(\phi)^{D(\delta)D(s)} \equiv D(\phi)^{D(\delta s)}$$

leaving one to show that the property holds for the individual cases and any wff $\psi(\vec{x})$ $D(\psi^\delta) = D(\psi)^{D(\delta)}$ and $D(\psi^s) = D(\psi)^{D(s)}$. The first follows by the fact that if y is a variable in ψ then $\psi \wedge (y \leftrightarrow z) \equiv \psi[z/y] \wedge (y \leftrightarrow z)$. The second follows by considering that s and $D(s)$ have the same effect on ψ.□

3.5 Graphs, sets and formulas

In the following we see how graphs can be constructed from formulas (as clause systems) and how formulas can be constructed from either sets or graphs such that a correspondence is preserved linking group action on the structures.

3.5.1 Graphs from formulas

The construction of this section is a more general interpretation derived from versions given by Crawford in [Cra92] and Boy de la Tour and Demri in [BdlT96b]. We consider here only clause systems of fixed size $k \geq 2$ although versions can easily be constructed to account for $k = 1$ by adding extra edges.

We construct graphs from $\alpha, \beta \subseteq \mathsf{Cl}_k(\vec{x})$ with $k \geq 2$ and outline some properties relating group action of $W(\vec{x})$ on $\mathsf{Cl}_l(\vec{x})$ to morphisms between the constructed graphs. Let $\mathsf{Nodes}(\vec{x}) = \mathsf{Cl}_k(\vec{x}) \oplus \mathsf{Lit}(\vec{x})$, the disjoint union of the clause sets and the literals. Then the group $S(\mathsf{Nodes}(\vec{x}))$ has a natural action on edges by pointwise action on the vertices. We construct the directed graph $\mathsf{Graph}_S(\alpha)$ by simply linking each literal-node to any clause-node containing that literal. Define

$$\mathsf{Graph}_S(\alpha) \stackrel{\text{def}}{=} \{\langle l, c\rangle \mid c \in \alpha, l \in c\} \cup \{\langle x, \neg x\rangle \mid x \in \vec{x}\}.$$

This graph is designed to have the property that its automorphisms correspond to stabilisers in the symmetric group $S(\vec{x})$ of the input clause system. We can easily adapt it to capture the stabilisers in $W(\vec{x})$ by putting a complementary edge between each pair of complementary literals, defining

$$\mathsf{Graph}_W(\alpha) \stackrel{\text{def}}{=} \mathsf{Graph}_S(\alpha) \cup \{\langle \neg x, x\rangle \mid x \in \vec{x}\}.$$

The first three statements in the following lemma assert that: i) the graph construction is monotonic; ii) if g is an injection node-wise of one construction into another, then g must also be induced by some element of the symmetric group acting naturally on $\mathsf{Nodes}(\vec{x})$; iii) the stabiliser of the clause system is the automorphism group of the constructed graph, with the appropriate natural actions implicit. The last statement asserts that these properties hold with respect to the $W(\vec{x})$ group.

Lemma 3.5.1 The following properties hold:

i). $\mathsf{Graph}_S(\alpha) \subseteq \mathsf{Graph}_S(\beta) \Leftrightarrow \alpha \subseteq \beta$.

ii). Let $g \in S(\mathsf{Nodes}(\vec{x}))$, the symmetric group over the vertices. Then

$$\mathsf{Graph}_S(\alpha)^g \subseteq \mathsf{Graph}_S(\beta) \Leftrightarrow g \in S(\vec{x})[\mathsf{Nodes}(\vec{x})].$$

iii). $\mathsf{Stab}(\alpha, S(\vec{x})) = \mathsf{Aut}(\mathsf{Graph}_S(\alpha))$.

iv). The above properties hold when replacing Graph_S with Graph_W and $S(\vec{x})$ with $W(\vec{x})$.

Proof For i) and ii) we observe that the graph construction ensures that the literal nodes are structurally distinguished from the other nodes and so maps must preserve the literal/clause relationship. Furthermore, the pairing of the literals ensures that complementarity is preserved. Then iii) follows easily from ii), and iv) is also straightforward.□

Example 3.5.2 The graph $\mathsf{Graph}_S(\alpha)$ for $\alpha = \{\{x, \neg y\}, \{y, \neg z\}, \{z, \neg x\}\}$ is illustrated in Figure 3.1. The automorphism group of this graph is the group on the vertices induced by the cyclic group on three variables which is therefore the same as the stabiliser of α in $S(x, y, z)$. Clote and Kranakis observe that a boolean function on three variables cannot have symmetry group C_3 in S_3 (see §2.9.2). It must have symmetry group S_3 if it is invariant under C_3. Therefore from knowing that the automorphism group of the graph construction corresponds to $C(x, y, z)$ we know that the syntactic symmetry

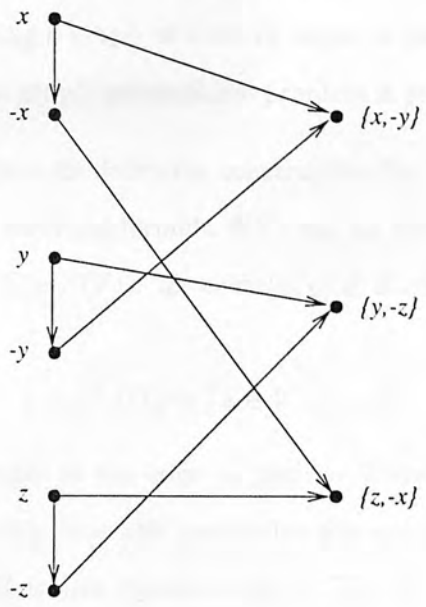


Figure 3.1: The graph $\mathsf{Graph}_S(\alpha)$ for $\alpha = \{\{x, \neg y\}, \{y, \neg z\}, \{z, \neg x\}\}$

group of the formula in $S(x, y, z)$ is $C(x, y, z)$ and hence its semantic symmetry group is $S(x, y, z)$.□

3.5.2 Formulas from sets

Given a set T and $s, t \in T$ we consider how to construct wffs $\mathsf{Form}(s), \mathsf{Form}(t)$ over T such that a correspondence can be established between actions of $S(T)$ on T and the constructed formulas. Define

$$\mathsf{Form}(s) \stackrel{\text{def}}{=} \bigwedge s.$$

Lemma 3.5.3 The following hold:

i). $\mathsf{Form}(s) \models \mathsf{Form}(t) \Leftrightarrow s \subseteq t$

ii). Let $g \in S(T)$. Then $\models \mathsf{Form}(s^g) \leftrightarrow \mathsf{Form}(s)^g$.

3.5.3 Formulas from graphs

We mention two methods of constructing a formula from a graph. The first was suggested by Kisielewicz [Kis98] in connection with the issue of representability §2.9.2.

The second method, viewing a graph as a set of edges, is essentially the method used by Luks to show that that graph isomorphism problem is reducible to OPS [Luk82].

Method 1 Kisielewicz gave the following construction for an undirected graph $H = (V, E)$ such that that the resulting formula $\phi(V)$ had an identical symmetry group to the graph, i.e. $\mathsf{Aut}(G) = \Sigma(\phi, S(V))$: for each $(x, y) \in E$ construct the minterm over V

$$x \wedge y \wedge (\bigwedge\{\neg z \mid z \in V \setminus \{x, y\})$$

with just the two end points of the edge as positive literals. Take $\mathsf{Form}(H)$ as the disjunction of these minterms. Now this means that any group representable as $\mathsf{Aut}(H)$ for an undirected graph H is also representable as $\Sigma(\phi, S(V))$ for a boolean formula $\phi = \mathsf{Form}(H)$ ([Kis98, Theorem 2.1]).

Method 2 Since the edges of a directed graph $H = \langle V, E\rangle$ are also just a *set* of objects $E \subseteq V^2$ then we can also use the formula construction of the previous section to represent a graph as a simple formula. Of course in this case the groups considered in connection with the construction are not arbitrary subgroups of $S(V^2)$ but specifically subgroups of $S(V)[V^2]$, i.e. those that act on the set of edges as $S(V)$ acts on vertices.

3.6 Tractable permutation group problems

Permutation groups can have large numbers of elements even if they are generated by just a few permutations. In this thesis we deal with (subgroups of) $W(\vec{x})$ which has order $n!2^n$ where $|\vec{x}| = n$. Furthermore, this group can be generated by just three permutations. The complexity problems of this thesis for which groups are part of the instances always use the generator representation as an input encoding, so this raises the question of what can be tractably asked about these permutation groups given such a representation. Several useful problems concerning permutation groups have solutions which run in polynomial time with respect to the degree of the group, a fact essential for many of the reduction and membership proofs of this thesis. Hence we review the main points in this section.

3.6.1 Preliminaries

The algorithms depend upon a basic tool in computational group theory, the stabiliser chain due to [Sim70, Sim78]. Let $G \leq S(\vec{x})$ where $\vec{x} = x_1, \ldots, x_n$. Denote by G_i the subgroup of G whose elements do not move any point in $x_1, \ldots, x_i$. Then we obtain a chain of subgroups

$$1 = G_{n-1} \leq \ldots \leq G_1 \leq G_0 = G.$$

Since the the size of each right transversal $[G_i : G_{i+1}]$ is less than or equal to $n-i$ then we can store all the transversals $C_i = G_i : G_{i+1}$ using a total of $O(n^2)$ permutations. This collection of transversals forms a generating set of permutations for G called a strong generating set.

3.6.2 Properties

A polynomial time algorithm for computing these transversals was given in [FHL80] and established the following theorem.

Theorem 3.6.1 [FHL80] Given a set of generators for $G \leq S(\vec{x})$, one can determine in polynomial time the order of G and whether a given permutation g is in G.

The order of G is given by $|C_0||C_1|\ldots|C_{n-2}|$ and indeed G itself is given by $C_0C_1\ldots C_{n-2}$. An unsophisticated and yet polynomial time algorithm for testing membership $g \in G$ involves generating the stabiliser chain for each of G and $\langle G, g\rangle$ and testing whether the two groups have the same order. The inductive nature of the stabiliser representation admits the following more elegant approach. Testing for $g \in G = G_0$ proceeds by selecting a permutation h from C_0 which sends x_1 to x_1^g. If no such element exists them membership fails. Otherwise we proceed to test for membership of gh^{-1} in G_1 and so on until we have $\epsilon \in G_{n-1}$, or failure.

The ability to test group membership in polynomial time implies that the test $J \leq G$ can also be performed in polynomial time for groups given as lists of generators. Simply

test membership in G for each generator j of J. Equality of groups given by different lists of generators can then be tested by checking inclusion in both directions.

Example 3.6.2 The stabiliser chain for the symmetric group $S(\vec{x})$ consists of $S(\vec{x})_i = S(x_{i+1}, \ldots, x_n)$ and the transversals C_i have size exactly $n-i$. An example transversal is just the generators normally associated with $S(x_{i+1}, \ldots, x_n)$, that is $(x_{i+1}\ x_{i+2}), \ldots, (x_{i+1}\ x_n)$.

3.7 Summary

Five transformation techniques concerning formulas, graphs, sets and groups have been described. These transformation techniques will be used to manipulate problems instances in Chapters 5–7 in order to exhibit reductions and membership proofs for the relevant complexity classes. In the following chapter, which looks at methods of exploiting symmetry in search, some of these techniques will also be relevant in the context of previously established complexity results.

Chapter 4

Symmetry Methods

4.1 Introduction

We look at ways in which symmetry can be used to simplify problems in the propositional domain. The format of this chapter will be to examine five papers from the automated reasoning literature covering different aspects of symmetry exploitation. We outline the main techniques employed, give examples and summarise the results that have been obtained. In a subsection for each paper, we discuss complexity issues associated with the work, pointing out any known complexity observations that have subsequently been made in connection with the work, including those of this thesis.

We present also two methods of exploiting the structure of groups in search, with some experimental results.

The use of symmetries as a method of simplifying search is quite old. As far back as 1874 symmetry is taken advantage of in a paper by Glaisher [Gla74] on the 8-queens problem. In mathematical proofs one often uses an arbitrary element of a set to construct an argument and then an appeal to symmetry in order to avoid repeated independent derivations that are simply permutational variants of one another. Using resolution as a base proof system for the propositional calculus, Krishnamurthy [Kri85] showed how certain tricky mathematical arguments could be encoded as short formal proofs in the resolution system augmented with principles of extension and symmetry. This paper, discussed below in §4.2, seems to have been the first presentation of sym-

metry principles in formal reasoning terms and has provided a basis for two further papers discussed in this chapter.

The first is by Benhamou and Sais [BS94]. While adopting the same underlying motivation of finding short proofs of propositional problems with symmetries, these authors address two issues not covered by Krishnamurthy's paper: establishing uniform methods of the application of symmetry principles which are not tied to particular problems; and efficient automated detection of symmetries to be used in the arguments. This paper is discussed in §4.5. Generalising Krishnamurthy's approach in a different direction, Boy de la Tour [BdlT96a] observes that the syntactic symmetries of a system of clauses do not provide all the symmetries that are applicable in arguments such as those of Krishnamurthy and of Benhamou and Sais. Furthermore, the shift to semantic symmetries permits a more systematic approach to the problem of saying how symmetries are preserved or lost under various manipulations of the problem. Tractable methods of finding some of these semantic symmetries are proposed and a generalisation of the Benhamou and Sais resolution method is obtained by considering these new symmetries. This paper is discussed in §4.7.

Two approaches where computational group theory has played a prominent part are also surveyed. The first is the backtrack search technique of Brown, Finklestein and Purdom [BFP88]. Their aim is to provide a general algorithm for searching in the presence of symmetry, which is not specific to the propositional domain. They take advantage of fast permutation group algorithms, some proposed by themselves, to solve the problem of finding a set of pairwise inequivalent solutions with respect to some group. We give an extended example in the relevant section for this paper §4.4 which attempts to relate the underlying principles of their approach to the heuristic method employed by Benhamou and Sais. An approach to symmetry exploitation which differs in nature from the idea of adapting search algorithms is the symmetry-breaking technique of Crawford, Ginsberg, Luks and Roy [CGLA96]. They argue that building symmetry methods into a particular search procedure is a somewhat fragile methodology. A better approach might be to preprocess the problem in the context

of some group of symmetries. The new problem could then be passed to any suitable search technique, for instance randomised hill-climbing techniques where there does not appear to be any obvious way of using symmetries dynamically. This paper is surveyed in §4.6.

In two final sections we examine methods for exploiting symmetry groups in search. The first, in §4.8 presents a variant of the semantic evaluation technique of Benhamou and Sais which uses an initial known group of symmetries rather than relying on local methods of determining symmetry. We show experimentally that a heuristic utilising the structure of the group can lead to smaller search spaces. In §4.9 we apply a similar technique to use of symmetry in finding better orderings for ordered binary decision diagrams.

The titles of the following five main sections are taken from the papers discussed. A convention adopted is that referenced theorems, sections etc. of the paper in question will be put in square brackets without an attached reference to the paper, which is to be understood from the title of the section.

4.2 Short proofs for tricky formulas

This section takes its title from the 1985 paper by Krishnamurthy [Kri85]. Krishnamurthy demonstrates short proofs of tricky combinatorial problems expressed as propositional tautologies. The paper discusses and compares two principles as augmentations of resolution. The first is the principle of extension, suggested by Tseitin [Tse68]. Krishnamurthy introduces the principle of symmetry, which we review in the next section.

The principle of extension allows a new propositional variable y to be created which is then defined in terms of variables already existing in a proof. A set of clauses is added to encode $y \leftrightarrow \phi(\vec{x})$ where $\vec{x}$ may contain new variables previously defined as well as the original variables of the problem. The objective of this is to be able to manipulate the defined variables instead of the formulas they stand for so as to significantly reduce

the length of a proof. Resolution extended with this principle (ER) has been shown to be sound.

4.3 Methods

The SR-I symmetry rule is defined in [Kri85] as follows

> Let C be a clause and σ a permutation of the set of variables occurring in C. Define $\sigma(C)$ in a natural way (i.e. the clause obtained by applying σ to each variable in $C \ldots$) For a set S of clauses, define σS as $\{\sigma C \mid C \in S\}$. Let Σ_S be the group of permutations of the variables in S that leaves S invariant, i.e. for all σ in Σ_S, $\sigma(S) = S$. The rule of symmetry allows the following derivation:
>
> $$\frac{F \quad \sigma \in \Sigma_S}{\sigma F} \tag{4.1}$$

meaning that if F is derivable in the context of S then so is σF. In our notation then the property that $\sigma \in \Sigma_S$ corresponds to $g \in \mathsf{Stab}(\alpha, S(\vec{x}))$ where $\alpha \subseteq \mathsf{Cl}(\vec{x})$.

Krishnamurthy proves [Lemma 2.1] by induction on proofs that the SR-I rule is sound with respect to resolution proofs

Krishnamurthy proposes a stronger rule, the SR-II rule, which exploits the fact that in a derivation $S \vdash C$ not all clauses of S may be required. If $A \subseteq S$ are the clauses used in a proof and σ is any permutation such that $\sigma(A) \subseteq S$ then $S \vdash \sigma(C)$ is also a derivation.

Example 4.3.1 Source in acyclic graph.

> **Fact 3.1**[p. 257] Every finite transitive digraph with no two-cycles must have a source.

The reasoning where symmetry is employed to show a refutation of the negation of Fact 3.1 is expressed as follows [p. 259]:

> Let us pick a vertex from $\{1, 2, \ldots, n\}$. Without loss of generality, assume it is n. .... Since n is not a source, there is an edge from some vertex to n, say from $n-1$. Continuing this argument we must either exhaust our vertex set, or come upon a cycle. In either case we arrive at a contradiction.

Krishnamurthy demonstrates a formal SR proof which imitates this intuitive 'without loss of generality' argument and shows that it has length $O(n^3)$ [Theorem 3.5]. The symmetry group in question here is the group induced on the set of propositional variables $x_{i,j}$ (asserting that there is an edge from i to j) by the symmetric group S_n acting on the vertices $\{1, \ldots, n\}$.

4.3.1 Results

As well as the above result for the transitive digraph problem, Krishnamurthy demonstrates SR proofs for tautologies derived from a weak upper bound on Ramsey's theorem: for every pair of integers $r_1, r_2 \geq 2$ there exists a sufficiently large integer n such that every undirected graph on n vertices contains either a clique of size r_1 or an independent set of size r_2. He encodes as a propositional formula the property that a particular function on r_1, r_2 giving a weak upper bound for n has the Ramsey property and uses the SR system to demonstrate $O(n^4)$ length proofs [Theorem 5.11].

Krishnamurthy poses the question of whether either of the two principles of extension and symmetry can simulate the other. He leaves it as an open question as to whether extension can simulate symmetry. It seems unlikely that the opposite is true since problems may be amenable to extension without containing any symmetry. Both rules could be combined. He distinguishes between the possibilities of allowing and disallowing the permutation of extended variables. In his checkerboard example (two opposite corners are removed from an $n \times n$ board and it becomes impossible to tile the board with dominoes), extension gives polynomial sized proofs [Theorem 4.7] but symmetry only a linear reduction in proof size of at most a factor of four, corresponding to the meagre number of (syntactic) symmetries associated with the problem. He suggests using the SR-II rule may give more power here. He argues that in cases where

both rules could be beneficially applied that symmetry may be a more intuitive way of thinking about what may amount to equivalent arguments.

4.3.2 Complexity issues

One of the high-level motivations for looking for short proofs of validity or unsatisfiability of propositional formulas is the question of whether the class NP is closed under complement. If it is not, as still seems to be the prevailing view, then this implies there are valid formulas for which short resolution proofs of validity cannot exist. Krishnamurthy's approach, not intended to be interpreted as an attempt to refute the NP $\neq$ co-NP question, is to study the limitations of resolution augmented with symmetry principles, arguing that attempts to establish lower bound results on the complexity of propositional proofs systems in general should address systems such as SR and ER. A proof of the fundamental intractability of the pigeonhole problem (Example 4.5.1) for resolution was given in [Hak85], coinciding with the publication of Krishnamurthy's paper. We meet in this chapter several ways in which symmetry rules can give rise to polynomial sized proofs of this problem, validating Krishnamurthy's original motivations.

From a more practical point of view there is the question of finding symmetry in the formulas to be used in Krishnamurthy's rules. Krishnamurthy did not address this problem as his main point was in demonstrating the short proofs as opposed to mechanising them. For this purpose one can extract a suitable group from an understanding of the problem from which the formula is extracted. Crawford appears to be the first to have shown how computing syntactic symmetries of a clause system can be reduced to finding automorphism group of a graph [Cra92]. The latter problem is not known to be tractable in general, although we meet in §8.3 some of the evidence that it is strictly easier than satisfiability checking itself. A later paper by Boy de la Tour and Demri [BdlT96b] independently showed the same result and also showed that finding suitable maps to apply in the SR-II rule was an NPC problem, by transformation from SUBGRAPH ISOMORPHISM.

4.4 Backtrack searching in the presence of symmetry

This section takes its title from the 1988 paper by Brown, Finklestein and Purdom [BFP88]. The authors show how techniques from computational group theory can be applied to improve the speed of backtrack searching on problems with symmetry. In the context of a group G, under which the candidate solutions of the problem are known to be invariant, the aim is to avoid searching equivalent portions of the search space with respect to the group, and to find via backtracking a set of solutions which are inequivalent with respect to the group. Their approach is a general one aiming to combine the underlying techniques used in specialised backtrack algorithms with fast permutation group algorithms for testing equivalence of search points. A restricted version of the general principle of their algorithm is analysed in the example below. We will then point out the more general features of their algorithm.

4.4.1 Example

We look at a propositional example where each variable may take one of the two truth values. The object is to find the set of inequivalent assignments to four variables $\vec{x} = x_1, \ldots, x_4$ with respect to the cyclic group $C(\vec{x}) = \langle g \rangle$ where $g = (x_1\ x_2\ x_3\ x_4)$. We could combine this with a search to determine satisfiability of a formula $\phi(\vec{x})$ known to be invariant under $C(\vec{x})$ in the following sense: at each point of the search we maintain a partially evaluated formula according to the values assigned to the variables in the search process; at some points we may be able to detect that the partially evaluated formula is in fact equivalent to 0 and so no satisfying assignments can be found beneath that point. The search tree could then be pruned at that point.

Furthermore, since we know that ϕ is invariant under $C(\vec{x})$ then we know that its set of models in $\mathsf{Asg}(\vec{x})$ consists of complete orbits under $C(\vec{x})$ and therefore it is satisfiable if and only if it is satisfied by some representative from one of these orbits. Hence we need to generate only leaves of the tree corresponding to a representative from each possible orbit. Therefore this satisfiability technique is complete when combined with

the group-theoretic approach. In the example below we examine just the underlying principle of searching a solution space in the context of a group of symmetries. We

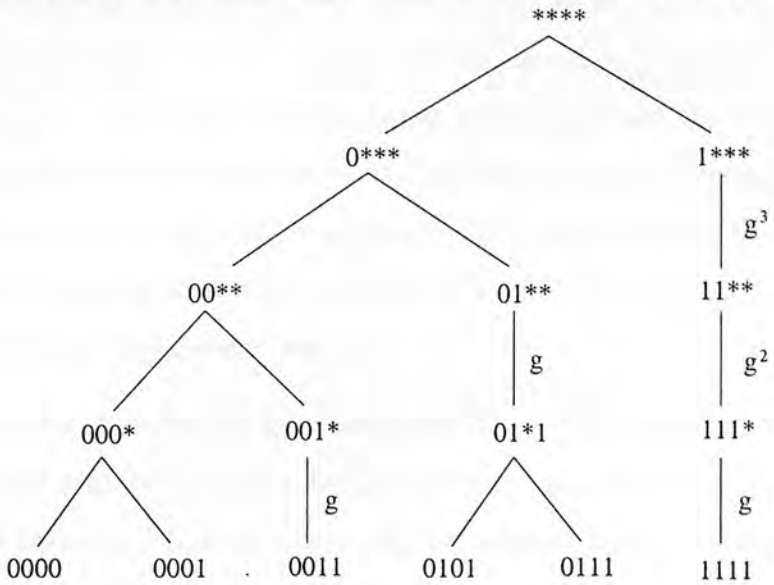


Figure 4.1: Search for distinct assignments under the cyclic group $C(\vec{x}) = \langle g \rangle$ on four points. An uninstantiated variable is forced to take a value if taking the opposite value would lead to a partial assignment that can be mapped by an element of the group to one which is strictly smaller under all remaining interpretations. The permutations in this case are indicated as edge labels.

settle the question of which representative to choose from each orbit by defining an ordering $\leq$ on assignments by lexicographic ordering on the string representations of $\mathsf{Asg}(\vec{x})$ with respect to the variable ordering $\vec{x}$. We choose the minimal element of an orbit under this ordering to represent the orbit. For example the representative of the orbit containing 0100 would be 0001, obtained by applying g^2.

Figure 4.1 indicates the search to find the minimal elements of each orbit induced by the action of the group $C(\vec{x})$. In the search, the four variables are initially uninstantiated. The aim is to prune away any parts of the tree that would lead to non-minimal valuations relative to the group $C(\vec{x})$ and the lexicographic ordering on the assignments. Consider the subtree at 1***. Observe that if the second variable is fixed at 0, then a single left rotation g^3 applied to 10** yields 0**1. In whichever way the two remaining variables in 10** are fixed, a rotation of the resulting string will be strictly smaller. That is 0**1 $<$ 10** for any realisation. Therefore, all leaves of 10** would be

non-minimal. The second bit is forced to be 1, and the third and fourth bits likewise.

For a partially instantiated string s we therefore look to see if setting a variable x_i to value v would mean that there is a $h \in C(\vec{x})$ such that $s^h < s$ for all possible realisations of s. This branch is then pruned and v must take the opposite value. Forcing of variables is a multi-phase process. Once some variables are restricted, it may be possible to restrict more variables as a result: this occurs in the rightmost branch of the figure. The pruning obeys two requirements: it removes only non-minimal leaves; it removes all the non-minimal leaves.

It may seem that choosing the *least* assignment to represent each orbit means that variable forcing will always involve fixing bits to 1. Consider the group $G = \langle g, f_4$ where g is as before and f_4 is the map which interchanges all four variables with their complements (see §2.8) and consider the search point 01**. Forcing sets the third bit to be 0 since if it were 1 then 011* maps to 100* by f_4 and then to 00*1 by g^{-1}. This last is now strictly smaller than 011* for any realisation. Thus variable forcing may coerce bits to be 0 as well as 1.

4.4.2 Enhancements

The above example and analysis arose out of an attempt to find some connections between techniques examined later in this chapter, namely variants of a heuristic proposed by Benhamou and Sais [BS94] and the symmetry-breaking technique of Crawford et al. [CGLA96]. We will refer to this example in the context of these other techniques. Later it was realised that the technique of the example is in fact just a restricted version of the general backtrack algorithm of [BFP88] to which we now return.

One generality they provide in the algorithm is that variables may range over some finite set of size m instead of just the set $\{0, 1\}$. The most serious restriction on the above example is that the ordering is fixed as the fixed lexicographic ordering throughout the search. The next variable to be used must be the next uninstantiated one in the ordering. Using 'dynamic search rearrangement' this restriction may be lifted. This means that the choice of the next variable to be used at any point can be

determined by some heuristic process such as the variable with the fewest remaining possible values. The algorithm must keep track of these ordering choices to ensure that the leaves of the search tree are still inequivalent under the group and include a representative of each orbit.

This incurs a corresponding cost in terms of having to recompute a stabiliser chain for the group G relative to the new chosen ordering at each search point. The importance of having a stabiliser chain for G based on the same ordering as the variables already instantiated at any point in the search is for the sake of the efficiency of their *Symtest* routine, which does the hard work of checking whether some partial instantiation maps to a smaller one under the group.

4.4.3 Results

The authors point out that whereas in practice the exploitation of symmetry may speed up searching greatly, there is no guarantee of speed up on any particular problem and the method does not improve the worst case time for searching. They tested the algorithm on the n-queens problem, comparing dynamic search rearrangement with and without the symmetry method, with results indicating considerable time and space improvements with symmetry techniques.

4.4.4 Complexity issues

The efficiency of the algorithm is dependent to a large extent on the efficiency of the various computational group theory procedures that have to be applied in the search. Their *Symtest* routine can take exponential time in the worst case. The problem solved by *Symtest* is at least as hard as the setwise stabiliser problem, for which no efficient procedure has yet been discovered. The problem mentioned above of computing a stabiliser chain with respect to a new ordering is considerably easier than computing the representation from scratch. They implemented an $O(n^3)$ algorithm for doing this where n is the number of points to be changed, improving previous methods.

4.5 Tractability through symmetries in the propositional calculus

This section takes its title from the paper by Benhamou and Sais [BS94]. The paper studies the use of symmetries in propositional calculus to improve the efficiency of automated deduction methods. The paper addresses problems not covered by Krishnamurthy's observations on how to construct short proofs, namely the detection of symmetries and uniform methods of using the symmetries in various proof systems.

4.5.1 Methods

Their general methodology is in finding symmetry to apply individually at each search point rather than operating with some initial known group. The authors present a method of finding a nontrivial syntactic symmetry σ of a system of clauses S using a backtrack method augmented with a number of useful heuristics. Two literals l_1, l_2 are then symmetric in S if $\sigma(l_1) = l_2$. The underlying principle of applying the symmetry information is encoded as [Theorem 4.4] which asserts that a formula has a model where l_1 is true if and only if it has a model where l_2 is true. This allows pruning to be performed as illustrated in the following example for the resolution method SLRI [KK71] which attempt to progressively refute each literal of some chosen initial clause by resolution with the remaining clauses.

Example 4.5.1 The pigeonhole problem. This is the well-known problem of placing n pigeons in $n-1$ holes, clearly impossible. But expressed as a propositional formula it is not so clear how to establish that it is contradictory in time polynomial in n because without search guidance every assignment of pigeons to holes would be considered. The propositional representation of the problem is syntactically symmetric under a large group of permutations of the variables p_{nh} denoting that pigeon n is in hole h. The group in question is the group induced on the set of variables p_{nh} by the product of symmetric groups on n and h acting on the indices. This is the set of clauses for three pigeons and two holes (example taken from [p. 95]):

$$p_{11} \vee p_{12} \quad \neg p_{11} \vee \neg p_{21} \quad \neg p_{12} \vee \neg p_{22}$$
$$p_{21} \vee p_{22} \quad \neg p_{11} \vee \neg p_{31} \quad \neg p_{12} \vee \neg p_{32}$$
$$p_{31} \vee p_{32} \quad \neg p_{21} \vee \neg p_{31} \quad \neg p_{22} \vee \neg p_{32}$$

The tree of Figure 4.2 indicates how the resolution process proceeds in the presence of a symmetry σ which maps literal p_{11} to p_{12}. Failure indicates a backtracking point where 1 has been derived. Refutation indicates that the literal at the root of the branch is refuted. Since literals p_{11} and p_{12} are symmetric in the original problem then

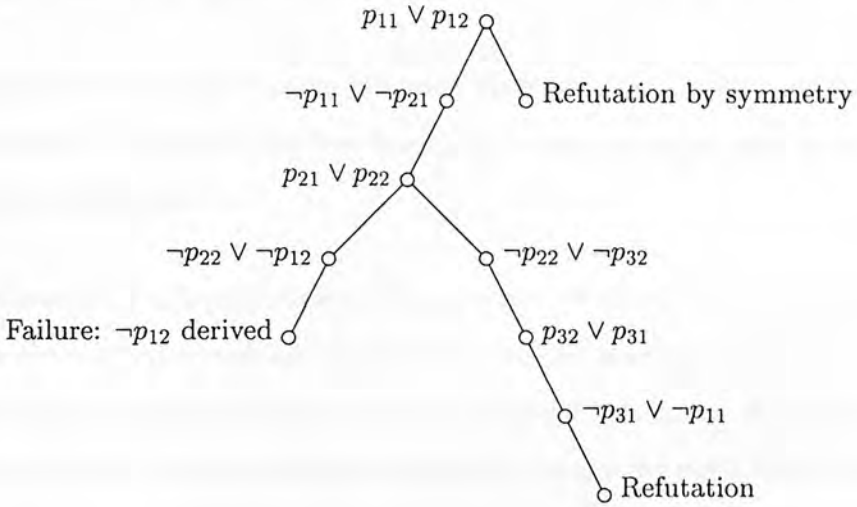


Figure 4.2: Resolution tree of the SLRI method for the 3-pigeons problem with symmetry [Figure 2]. The clause chosen for resolution is shown at each branch point.

a refutation of p_{12} leads to a symmetric refutation of p_{11}.□

They have further applied this principle in the semantic evaluation method, a divide and conquer strategy exploiting the fact the formula ϕ is equivalent to the disjunction of the two formulas obtained by evaluating any variable x of ϕ at 1 and 0 respectively. Denote by ϕ_x the formula obtained by setting variable x to 1, and by $\phi_{\neg x}$ the formula obtained by setting x to 0. Then for a list $\vec{l}$ of literals, $\phi_{\vec{l}}$ is the partial valuation corresponding the sequential evaluation of each literal as above. Their observation is that if G is a symmetry group of a formula ϕ then an enhanced version of the semantic evaluation principle applies [§6.3]: ϕ is satisfiable if and only if [ϕ_x is satisfiable] or [$\phi_{\vec{l}}$ is satisfiable], where $\vec{l}$ is the set $\{\neg l \mid l \in \mathsf{Orb}(x, G)\}$: that is, the set of negations of

literals which are in the orbit of x in G.

To see this, assume ϕ is satisfiable and ϕ_x is not satisfiable. Then for any assignment $a \in \mathsf{Asg}(\vec{x})$, $\phi(a) = 1$ implies that $\neg x(a) = 1$. But if a is model of ϕ, this means that a^g is also a model of ϕ for all $g \in G$ (corresponding to [Proposition 4.1]). Therefore $\neg x(a^g) = 1$, or by definition $(\neg x)^{g^{-1}}(a) = 1$ and so in any model of ϕ the set of negations of literals which are in the orbit of x in G must all take the value 1. Fixing a lot of variables at the same time has the effect of pruning a large proportion of the models that would otherwise be examined.

Example 4.5.2 (The Ramsey colouring problem) This is the problem of colouring the edges of a complete graph of n vertices with c colours so that no monochromatic triangle is formed.□

A refutation of this problem for the case with $n = 17$ and $c = 3$ was extracted using semantic evaluation methods augmented with the above pruning rule. The formula which encodes the problem has 408 variables and 2,584 clauses. No machine refutation of this problem had been previously exhibited, though the result is well-known through analytical methods.

4.5.2 Results

In the pigeons example above the authors give results for their SLRI implementation indicating refutations of size $O(n)$ and taking time $O(n^2)$ where n is the number of pigeons. In the semantic evaluation method they have successfully shown a refutation in 30 minutes CPU time on a Sun 4 of the problem of Ramsey with seventeen vertices as indicated above as well as substantial results for other combinatorial problems.

4.5.3 Complexity issues

One feature of their approach to utilising symmetries is that their algorithms for the various search procedures invoke a detection method at each point of the search tree. This has a theoretical advantage in that methods for exploiting an initial known group

G of symmetries must update the group as the search progresses and may be reduced to the trivial group at a point where more symmetries still actually exist. A variant of their semantic evaluation technique which works on this basis is examined §4.8. The disadvantage of their approach is in the computational cost of detecting a nontrivial symmetry at every search point. On the other hand their detection method seems to be quite successful and they indicate that in problems with a lot of symmetry they can find a nontrivial symmetry in linear time in general. Their detection method has a timeout mechanism which prevents more than a certain amount of backtracking to be performed in searching for symmetries so that if no symmetry can be found within that time the search can progress without symmetry. Boy de la Tour and Demri showed in [BdlT96b] that the technique Benhamou and Sais employ is unlikely to have a better complexity in general than the GRAPH ISOMORPHISM problem and would in the worst case require exponential time (without timeout) according to current best upper bounds on the complexity of this problem.

4.6 Symmetry-breaking predicates for search problems

This section takes its title from the paper by Crawford, Ginsberg, Luks and Roy [CGLA96]. The *symmetry-breaking* technique is used as a preprocessing technique exploiting symmetries $G \leq \Sigma(\phi)$ of formula ϕ. A symmetry-breaking predicate $\mathsf{Break}(G)$, which is itself another formula, is added by conjunction to ϕ preserving satisfiability. The required property of $\mathsf{Break}(G)$ is that it is has at least one model, and preferably just one, from each equivalence class induced by G on $\mathsf{Asg}(\vec{x})$ where $\vec{x} = \mathsf{Vars}(\phi)$.

If ϕ is invariant under G then its models are divided into whole orbits of $\mathsf{Asg}(\vec{x})$ under G. Therefore ϕ is satisfiable if and only if it is satisfied by any representative from one of the orbits. By adding $\mathsf{Break}(G)$ to a formula with symmetry G one reduces the models of the resulting formula to representatives of the G-orbits of models of ϕ. Thus $\mathsf{Break}(G) \wedge \phi$ is satisfiable if and only if ϕ is satisfiable. Furthermore certain search algorithms will benefit from the more constrained search space that the additional predicate provides.

The authors point out that the symmetry-breaking technique is a new symmetry method, differing from the methods seen so far in this chapter where the symmetry information is exploited by building symmetry techniques into the search algorithms. The authors argue that this is a good thing because of the danger of expending too much effort tying symmetry exploitation to a particular search mechanism in an environment where new techniques are being developed at a rapid pace.

Instead the preprocessing technique can be used as a front end to any satisfiability method. One advantage of computing $\mathsf{Break}(G)$ as a preprocessing method (where a reasonable symmetry breaker can be computed) is that the formula can be archived and retrieved to be applied to any problem with group G.

4.6.1 Methods

In the optimal case they would like to make $\mathsf{Break}(G)$ a formula whose models are unique representatives of the G-classes of $\mathsf{Asg}(\vec{x})$. For instance with $\vec{x} = x_1, \ldots, x_4$ and $C(\vec{x})$ the cyclic group of §4.4.1 then $\mathsf{Break}(C(\vec{x}))$ would be a formula whose models were precisely the leaves given in the tree of Figure 4.1. One possible realisation of $\mathsf{Break}(C(\vec{x}))$ is the formula

$$(\neg x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_4) \wedge (x_1 \vee x_2 \vee \neg x_3).$$

Take as the formula below as an example formula with symmetry $C(\vec{x})$:

$$\phi \stackrel{\text{def}}{=} (x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee \neg x_3 \vee x_4).$$

The formula ϕ has 14 models over $\vec{x}$. Combining ϕ with the $\mathsf{Break}(C(\vec{x}))$ yields a formula with 9 fewer models without affecting satisfiability.

Their method of computing $\mathsf{Break}(G)$ corresponds to constructing a propositional formula which is true of only the lexicographically least element of each G-orbit of $\mathsf{Asg}(\vec{x})$ (the ordering on assignments used in the search example of §4.4.1). They put $x \leq y$ to represent the formula $x \rightarrow y$. Using $\vec{x}_i$ to represent the initial sequence $x_1, \ldots, x_i$ of $\vec{x}$

then the formula representing $\vec{x} \leq \vec{y}$ is defined as

$$\vec{x} \leq \vec{y} \stackrel{\text{def}}{=} \bigwedge_{i=1}^{n} P_i(\vec{x}, \vec{y})$$

where $P_i(\vec{x}, \vec{y})$ represents the formula

$$\vec{x}_{i-1} = \vec{y}_{i-1} \rightarrow x_i \leq y_i.$$

The predicate that is true of only assignments which do have no larger image under group element g is then $\vec{x} \leq (\vec{x})^g$ which expands to a formula as below:

$$\begin{aligned} & & & (x_1 \rightarrow x_1^g) \\ & \wedge\ (x_1 \leftrightarrow x_1^g) & \rightarrow & (x_2 \rightarrow x_2^g) \\ & & \cdots \rightarrow & \cdots \\ \wedge\ (x_1 \leftrightarrow x_1^g) \wedge\ & \ldots (x_{n-1} \leftrightarrow x_{n-1}^g) & \rightarrow & (x_n \rightarrow x_n^g). \end{aligned}$$

The symmetry-breaking predicate for group G is then

$$\bigwedge_{g \in G} \vec{x} \leq (\vec{x})^g. \tag{4.2}$$

If G is the set of syntactic symmetries of formula ϕ, they propose this predicate as the symmetry-breaking predicate for the formula ϕ [Proposition 3.1].

The method used for computing the syntactic symmetries of ϕ was first demonstrated by Crawford in [Cra92]. The hard work of this computation is performed efficiently by the *nauty* graph isomorphism/automorphism program of McKay [McK92] once the construction of §3.5.1 has been employed. They applied the symmetry-breaking technique in two case studies, the n-queens problem and the pigeonhole problem. Below we review these results. We look at the complexity issues first because the difficulty of computing the symmetry-breaking predicate is connected with the approximation techniques they have used in the experimental results.

4.6.2 Complexity issues

Despite demonstrating some simplifications of the construction of Break(G) based on group theoretic analysis of some redundancy in the basic representation (4.2) they show

that Break(G) is NP-hard to compute [Theorem 3.2] and therefore various approximations to the symmetry-breaking predicate are suggested. A partial symmetry-breaking predicate is defined to be one which is true of at least one representative of each G-orbit on the assignments. It is therefore an approximation to the requirement that exactly one must be true, and adding the approximate predicate still preserves satisfiability. A method they suggest for a partial symmetry-breaking predicate seems to correspond to weakening the full lexicographic ordering to a partial one on an initial sequence of $\vec{x}$. Another suggested approach is to calculate the fragments $\vec{x} \leq (\vec{x})^g$ for only the generators g of the group, a technique that gives good results in some cases.

For certain useful groups they show computing Break(G) is tractable. For instance, Break($S(\vec{x})$) can be simplified drastically to

$$\bigwedge_{1 \leq i \leq j \leq n} x_i \rightarrow x_j.$$

To see this, consider that the smallest elements of each $S(\vec{x})$ orbit on Asg($\vec{x}$) are simply those in which 0's occur to the left of 1's: with $n = 4$ one has the representative assignments

$$0000 \quad 0001 \quad 0011 \quad 0111 \quad 1111.$$

We show later that the problem of whether a formula $\phi(\vec{x})$ has the property that it is true on at least one member of each G-orbit on Asg($\vec{x}$) which corresponds to the question

> Is ϕ a partial symmetry-breaking predicate for group G?

is complete for the class Π_2^p. It is the complement of the G-ISF problem of §7.3.

4.6.3 Results

A prototype system was implemented and results were given for the n-queens problem and pigeonhole problem. The syntactic symmetries of the formula ϕ are computed using the method of [Cra92]. The symmetry-breaking predicate is computed by one of several

approximation methods except in cases where it is tractable to compute the entire predicate (for example with a small group or the above symmetric group example). The predicate is added to ϕ and the combined formula is passed to the TABLEAU algorithm [CA96], a fast Davis-Putnam derivative. They show that entire process is polynomial for the pigeonhole problem, including computation of the syntactic symmetry group of the formula. In this case they used just the generators for the symmetry group returned by the symmetry-detection algorithm to construct the symmetry-breaking predicate. In the n-queens problem it is tractable to generate the full predicate for the 8 symmetries of the problem. The extra costs involved in computing the predicate meant that the symmetry method did not improve on the underlying method before board size 22.

4.7 Ground resolution with group computations on semantic symmetries

This section takes its title from the paper of Boy de la Tour [BdlT96a]. Many of the notions and themes of this paper are to be found elsewhere in this thesis in particular in the section on symmetry §2.9 and in Chapters 5 and 6 in relation to the complexity aspects of semantic symmetries. We therefore cover here just the main points that are not explicit elsewhere.

The paper presents the notion of semantic symmetries which are described as symmetries of the set of models of a formula and defined with logical equivalence as the measure of invariance. A semantic symmetry of ϕ is therefore a map g such that $\phi^g \equiv \phi$ (§2.9). This is in contrast to the syntactic symmetries $\sigma(S) = S$ of a system S of clauses considered by the papers surveyed previously in this chapter (notions discussed in §3.2). Motivations for this shift include the conservation of semantic symmetries by inference steps such as adding an entailed proposition to the formula, and the fact that there are generally more semantic symmetries available than syntactic ones. The more symmetries that are known the shorter will be the proofs, derivations, search trees etc. that use symmetry methods.

Although it is shown by Boy de la Tour that in the general case it is intractable relative to satisfiability checking to even determine whether a given map is a semantic symmetry of a formula, he describes a method by which additional semantic symmetries which may not be symmetries of the clause system can be inferred by tractable processes in some cases. We look at this briefly in the next section. We have already seen one way in which more symmetries can be inferred than are available syntactically and that is when the syntactic group is known to be a non-representable group as in Example 3.5.2. A new technique is discussed in §9.2.1.

Boy de la Tour introduces a methodology for describing how the relationship between a known symmetry group of a formula changes under some manipulation of the formula. He uses rules of the form:

$$\phi : G \vdash \phi' : G' \quad \text{if } \textit{condition}$$

where the rule is correct if, when $G \leq \Sigma(\phi)$ and *condition* is true, then $G' \leq \Sigma(\phi')$ holds.

A technique called *symmetric factorisation* is presented which is viewed as a generalisation of the Benhamou and Sais resolution method of 4.5.1. As well as shifting to semantic symmetries, it seems to be a generalisation in that it can be used either as a preprocessing method or as a symmetry specialisation of the resolution search methods where it corresponds closely to the Benhamou and Sais technique. His described implementation uses the method as a preprocessor. We examine factorisation below.

4.7.1 Methods

A symmetry argument deriving new semantic symmetries in some cases is described. [Lemma 7] shows that for wff ϕ and $l_1, l_2 \in \mathsf{Lit}(\mathsf{Vars}(\phi))$, if $\phi \models l_1 \leftrightarrow l_2$ then the map $(x\ y)^\neg$ must be a symmetry of ϕ. (In the case that $y = \neg x$ then ϕ must be contradictory and so all symmetries can be inferred [Lemma 22].) [Lemma 5] shows that $\phi \models l_1 \leftrightarrow l_2$ can be derived from $\phi \models l_1 \rightarrow l_2$ if l_1, l_2 are in the same orbit of some known symmetry group $G \leq \Sigma(\phi)$. It is then observed that more equivalences between literals can be

systematically derived by applying symmetries of ϕ to the equivalence $l_1 \leftrightarrow l_2$. This is all combined into a 'symmetric implication rule' [Theorem 9].

Boy de la Tour introduces the idea of factorising a formula ϕ relative to a known symmetry group G. This is somewhat akin to the symmetry-breaking technique in that the idea is to add constraints to the problem, hopefully leading to more direct proofs. Instead of being derived directly from the group as in the symmetry-breaking case, the extra constraints are determined by analysing the particular presentation of the problem.

Let $G \leq \Sigma(\phi)$ and let c be a clause in the CNF formula ϕ. Let c contain the literals l_1, l_2 which are in the same orbit of G. Then one of the literals can be deleted from the clause without affecting satisfiability of ϕ because ϕ has a model a making l_1 true if and only if it has a model a^g making $l_2 = l_1^g$ true. (This is the argument of [BS94, Theorem 4.4]).

This reasoning corresponds closely to the 'without loss of generality' intuition behind many of the symmetry arguments encountered already in this chapter. Given a disjunction of symmetric propositions as an assumption, one shows that, if a representative assumption is sufficient to establish a refutation or to construct a model, then corresponding refutations or models must follow from any other member of the disjunction.

A G-factor of a clause c is a defined to be a clause c' containing at least one representative literal from each G-orbit of the literals of c. The representatives do not seem to have to be in the clause c provided they are in the same G-orbits of literals that are in the clause. In the case that $c' \subseteq c$ then c' can *replace* c. In fact, this is just saying that c' can be *added* and then c removed by subsumption.

The factorisation method proceeds iteratively in the context of group G. If G is trivial then we have no factorisation possible. Otherwise choose a suitable factor clause $c \in \phi$. Compute a G-factor c' and add the factor to ϕ, optionally performing a subsumption check to remove clauses subsumed by the factor (subsumption preserves semantic symmetries [p. 488]). Then we have to adjust G to account for loss of

symmetry. Boy de la Tour shows that a suitable new group to consider is the setwise stabiliser $G_{\{c'\}}$ of the literals of the factor c' in G giving the 'symmetric factorisation rule'.

$$\phi : G \vdash (\phi \wedge c') : G_{\{c'\}} \quad \text{if } c' \text{ is a } G\text{-factor of a clause in } \phi.$$

If the new group is nontrivial then we can proceed to find and add more factors.

The factorisation method is representation dependent. It is a simple matter to construct a symmetric formula in which no clause contains two symmetric literals. Consider a group G with three orbits and a clause $c = l_1 \vee l_2 \vee l_3$ containing a literal from each orbit. The CNF formula $\bigwedge_{g \in G} c^g$ is symmetric under G but admits no factorisation. Boy de la Tour emphasises that although no factorisation may be possible at the beginning or some intermediate stage of a resolution proof it may become possible after some resolution steps which collect together some literals in the same orbit. Hence the method could be viewed as either a preprocessing method, where he shows a GAP implementation works extremely well in the pigeonhole problem, or as a hybrid approach combining preprocessing with 'dynamic' factorising.

4.7.2 Complexity issues

Boy de la Tour introduces the MSYMM and GSYMM complexity problems which are discussed in §5.2 and §6.5.

In the following two sections we examine further symmetry methods based around experiments conducted in the work of this thesis.

4.8 A variant of the semantic evaluation technique

It is possible in the semantic evaluation technique of Benhamou and Sais to exploit a previously known group G of symmetries of the original problem, instead of relying on a local search procedure to find a new symmetry at each branch point. The original group must be updated as one progresses through the search tree. As variables of the original formula are evaluated, symmetries of the resultant problem in the group G

will be lost but there is a systematic way of deriving a subgroup of G which will still be a symmetry group of the partially evaluated formula.

If $\vec{l}$ is a list of literals (possibly a singleton) then a symmetry group of $\phi_{\vec{l}}$ can be extracted from the known group G of ϕ so that the heuristic can be further applied at subsequent levels of the tree.

Lemma 4.8.1 If $H \leq W(\vec{x})$ is a group of symmetries of $\phi(\vec{x})$ and the set $\{\vec{l}\}$ is invariant under H then H is a group of symmetries of $\phi_{\vec{l}}$.

Given some initial group of symmetries G then the *pointwise* stabiliser $G_{\vec{l}} \leq G$ of the list $\vec{l}$ in the group G is a group and can be computed in polynomial time This is the same notion of stabiliser as in the discussion of stabiliser chain representations of permutation groups in §3.6. This subgroup $G_{\vec{l}}$ of G is a symmetry group of the formula ϕ after evaluation of the literals $\vec{l}$ by the above lemma since it leaves $\{\vec{l}\}$ invariant.

Similarly we could consider the larger *setwise* stabiliser $G_{\{\vec{l}\}} \leq G$ of the set $\{\vec{l}\}$ in G since this also fixes the set $\{\vec{l}\}$. In the general case no polynomial algorithm is known for computing the subgroup of a group acting on a set which leaves a given subset invariant. However, efficient implementations such as the GAP[1] function `Stabilizer` make it feasible to use the setwise stabiliser option in the variant of the semantic evaluation heuristic. We illustrate how it works in the following example.

Example 4.8.2 We apply the variant of the algorithm of Benhamou and Sais in the context of the backtrack search procedure of [BFP88] and the example examined above in §4.4.1. We look at the underlying search procedure in the context of symmetries.

At any search point we have a partially fixed assignment such as 0**1 and some applicable subgroup H of our original group G. Choose any unfixed variable x_i and a value $v \in \{0, 1\}$. Then we apply Benhamou and Sais's enhanced version of the semantic evaluation rule: set x_i to v in the left subnode. Set the orbit $\vec{l}$ of x_i in H to $1 - v$ in the right subnode. If the orbit of x_i in H contains $\neg x_i$ or if x_i has already been set to

[1] GAP [S+95] is a computer algebra system dedicated to computational group theory.

v then this branch can be pruned completely. Then we proceed in depth first manner to the two subnodes with applicable groups respectively H_{x_i} and $H_{\{\vec{l}\}}$.

We can apply this variant of the Benhamou and Sais heuristic to the same group $C(\vec{x})$ as the example of Figure 4.1 to compare the effects in terms of the number of leaves visited. The algorithm is flexible as to the choice of x_i and v at each node, but let us assume that x_i is always chosen to be the next unfixed variable in the left to right order and v is chosen to be 0.

At the top node **** the heuristic works the same as in the method based upon minimising the partial assignment. We fix the left subnode to 0***. In the right subnode the orbit of x_1 under $C(\vec{x})$ is all four points which then take the value 1. This cyclic group, however, is trivialised by fixing any point, or strict subset of the points. At level two, therefore, the symmetry is all used up and 9 of the 16 assignments are generated as opposed to the minimal number of 6 out of 16 in the lexicographic method.

Consider the group $D(\vec{x}) = \langle g, (x_1\ x_3)\rangle$, the dihedral group of the eight symmetries of the square. This group is generated by the rotation g of the four elements and a reflection along a diagonal. This group induces exactly the same orbits on $\mathsf{Asg}(\vec{x})$ as the cyclic group $C(\vec{x})$. This means that any formula stabilized by $C(\vec{x})$ is also stabilized by the larger group $D(\vec{x})$. In the parlance of representability discussed in §2.9.2 this is the same as saying that the group $C(\vec{x})$ is not representable as the semantic symmetry group of any formula $\phi(\vec{x})$.

Now the variant of the heuristic applied to this group yields an identical tree to that of Figure 4.1. So in fact the heuristic is optimal for this group: it finds exactly a set of inequivalent assignments under $D(\vec{x})$ and therefore uses the symmetry optimally in the sense of reaching only inequivalent search points under the group.

Consider the node 00**. The applicable subgroup $H < D(\vec{x})$ which fixes setwise the first two variables is the group $\langle (x_1\ x_2)(x_3\ x_4)\rangle$. The left subnode is then set to 000* and the right subnode to 0011, since the third and fourth variables are in the same

Group	Optimal	Longest	Random	Shortest
1 (C_{12})	352	2049	2044	2049
31	122	999	862	878
61	101	417	450	301
91	77	328	316	273
121	64	475	339	289
151	56	417	218	151
181	39	104	124	80
211	70	193	188	156
241	55	219	130	61
271	36	139	75	45
301 (S_{12})	13	13	13	13

Table 4.1: Numbers of leaves visited by variants of the Benhamou and Sais heuristic for various transitive subgroups of of degree 12.

orbit of the subgroup H.□

Choosing variables One general guiding heuristic suggested by Benhamou and Sais in the context of their methods is that in choosing which variable to evaluate next one should aim to preserve as much of the symmetry as possible at subsequent search points. This raises the question of how this heuristic could be applied uniformly in the context of the variant of their method discussed in this section. One observation is that symmetry can be preserved if notice is taken of the orbits induced on the variables by the group applicable to some search point. Recall that a group G acting on a set partitions the set into disjoint orbits. The setwise stabiliser of any of these orbits in G is G itself. So if it happened that the current evaluated formula $\phi_{\vec{l}}$ was such that $\vec{l}$ was a whole orbit of G then we would be have the whole of G to work with again.

A technique that suggests itself is to try and complete orbits of variables thus aiming to recover the whole group again. This seems to translate heuristically into choosing a variable from the smallest available orbit of the current known group.

Experiment To test this idea we implemented the construction of semantic evaluation trees using this variant of the Benhamou and Sais heuristic. That is we simply generate the whole tree as in Figure 4.1 for different initial symmetry groups and use the number of leaves of the tree as a guide to the success of the various parameters. Branches are pruned according to the Benhamou and Sais evaluation rule with the group derived from the initial known symmetry group as in the above discussion using the setwise stabiliser. The fewer leaves, the closer the heuristic is to the optimal situation of only visiting distinct points under the group.

In Table 4.1 we give numbers of leaves visited for a number of different cases. The independent variables are: the choice of group; the option of whether to choose the next variable from the longest, random or shortest orbit of the group applicable at a search point. The dependent variable is the number of leaves in the search tree resulting from these choices.

The groups of Table 4.1 are given as identification numbers corresponding to GAP's library of transitive groups of degree 12. Group 1 is the cyclic group on the 12 variables and group 301 is the symmetric group on 12 variables. The others are chosen uniformly from the 301 available groups of this degree. The results indicate that it was the correct choice to opt for the next variable from the smallest available orbit. The opposite choice leads to worse performance than choosing a random variable.

Conclusion We have illustrated that a well-characterised variant of the Benhamou and Sais semantic evaluation technique using symmetries can be viewed as an approximation to the optimal backtracking technique discussed in §4.4 which guarantees to visit only distinct points. A heuristic which chooses the next variable from the smallest remaining orbit of the symmetry group provides a method of obtaining smaller search trees in a way which is essentially independent of the problem.

In the following section we show how to apply the same kind of analysis to the construction of small OBDDs for formulas with known symmetry.

4.9 Orderings for OBDDs with symmetry

The main point of OBDDs (see Appendix A) is in identifying common subexpressions of functions. The canonicity property of OBDDs ensures that where such common subexpressions are found relative to an ordering, then they will be exploited. In other words, one never has two nonterminal nodes representing the same function for some residual sequence of variables. The presence of symmetry in a function provides a heuristic for determining variable orderings which aim to collect classes of such common subexpressions. We illustrate this with experimental results for groups of degree 12.

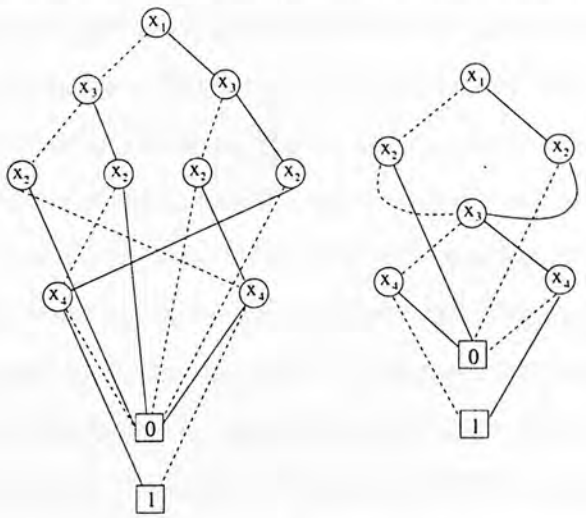


Figure 4.3: Large and small OBDDs for the function $f = (a \leftrightarrow b) \wedge (c \leftrightarrow d)$ (example from [And96]). See §A. Dotted lines are 0-arcs and plain lines are 1-arcs.

Consider the formula $\phi \stackrel{\text{def}}{=} (x_1 \leftrightarrow x_2) \wedge (x_3 \leftrightarrow x_4)$. Observe that $\phi_{x_1,\neg x_2} \equiv \phi_{\neg x_1,x_2}$ and $\phi_{x_1,x_2} \equiv \phi_{\neg x_1,\neg x_2}$. Hence if an ordering were chosen such that x_1, x_2 were the first variables to be evaluated then one would have only two instead of a maximum of four nodes at level 3. The maximum is realised if one chooses the ordering x_1, x_3, x_2, x_4. Figure 4.3 shows the two resulting OBDDs. These particular equivalences between the partially evaluated formulas can be derived through consideration of the symmetry group of ϕ, which is the group G generated by $\langle (x_1\ x_2)^\neg, (x_1\ x_3)(x_2\ x_4)^\neg, (x_1\ \neg x_1)(x_2\ \neg x_2)\rangle$.

The following lemma formalises this symmetry property in terms of subexpressions which can always be collected.

Lemma 4.9.1 Let $\phi(\vec{x})$ be a wff and let $G \leq \Sigma(\phi)$ Let $\vec{y} \subset \vec{x}$ and let $\vec{l}$ be a complement-free list of literals over $\vec{y}$. If $g \in G_{\vec{x}\setminus\vec{y}}$, that is if g stabilises pointwise all the elements in $\vec{x}$ not in $\vec{y}$, then $\phi_{\vec{l}} \equiv \phi_{\vec{l}^g}$: the formula ϕ partially evaluated at $\vec{l}$ is equal to the formula partially evaluated at the images of $\vec{l}$ under g.

In the example, let $\vec{y}$ be $\{x_1, x_2\}$. Then $\vec{x}\setminus\vec{y} = \{x_3, x_4\}$ and we have both $(x_1\ x_2)^\neg$ and $(x_1\ \neg x_1)(x_2\ \neg x_2)$ in G_{x_3,x_4}. The lemma then gives the two common subexpressions $\phi_{x_1,\neg x_2} \equiv \phi_{\neg x_1,x_2}$ and $\phi_{x_1,x_2} \equiv \phi_{\neg x_1,\neg x_2}$.

Ordering This suggests that a good ordering for an OBDD of a formula with a symmetry group G is one in which tail segments of the ordering (the sequences $\vec{x}\setminus\vec{y}$) induce nontrivial pointwise stabilisers. To get the largest number of common subexpressions, we require the stabilisers $G_{\vec{x}\setminus\vec{y}}$ to be as large as possible. In similar fashion to the idea of optimising the variant of the Benhamou and Sais heuristic, discussed in the above section, we select a point x in the smallest orbit of the group G, with the idea that, fixing this point, we should obtain a larger stabiliser than if we fixed a point in a larger orbit. We then proceed to choose the point x' in the smallest orbit (which may be a singleton) of the resulting group G_x and fix that next obtaining $G_{x,x'}$ and so on. This continues until the group is trivialised. Note that pointwise stabiliser computation is polynomial as discussed in §3.6. The inverse of this heuristic, where the largest orbit is chosen first should be a high-cost alternative and give larger OBDDs since we are effectively trying to minimise the benefit of Lemma 4.9.1. In the following we test this idea by looking empirically at what symmetry can say about OBDD size in general.

Upper bound on size Lemma 4.9.1 provides an upper bound on OBDD size in the context of an ordering $\vec{x} = x_1, \ldots, x_n$ and a wffs $\phi(\vec{x})$ with a symmetry group $G \leq W(\vec{x})$. There can be no more nodes at level i of the OBDD than the number of orbits in $\mathsf{Asg}(x_1, \ldots, x_{i-1})$ under the group $G_{x_i,\ldots,x_n}$: this is the group that fixes pointwise the last $n-i$ points. Hence the total size of the OBDD cannot be more than the sum of these upperbounds for the number of nodes at each level:

$$2 + \sum_{i=1}^{n} |\mathsf{Orbits}(G_{x_i \ldots x_n}, \mathsf{Asg}(x_1, \ldots, x_{i-1}))|. \tag{4.3}$$

where $\mathsf{Orbits}(H, S)$ computes the partition of the set S into orbits under group H. At the lowest level (where i would be $n+1$ above), instead of having $|\mathsf{Orbits}(G, \mathsf{Asg}(\vec{x}))|$, we have just the two terminal nodes. Hence the 2 in front.

As a specific example of a group, for the symmetric group on the literals, $S(\vec{x})$ the number of orbits induced by the pointwise stabilisers is linear in i. In fact it is always $i+1$, whichever ordering is chosen. Hence the upperbound for the OBDD is quadratic in n. This result for the symmetric group is standard (see [Bry92, p.7] for an argument based upon a constructed circuit).

Experiment In Table 4.2, we consider the case $n = 12$. The upperbounds on the OBDD size by (4.3) for a selection of different groups are given. As before we use a cross section of the 301 transitive groups of degree 12, taken from GAP's transitive group library. The Longest and Shortest columns give the upperbound OBDD size

Group	Group size	Longest	Random	Shortest
1 (C_{12})	12	4,097	4,097	4,097
61	96	1,961	1,823	1,007
121	216	1,617	1,281	541
151	384	1,565	1,138	449
181	720	1,001	984	881
211	1296	776	707	519
241	2304	891	705	362
271	7,776	567	498	273
280	15,552	552	511	272
285	23,040	565	463	185
290	41,472	369	352	213
300 (A_{12})	239,500,800	81	81	81
301 (S_{12})	479,001,600	80	80	80

Table 4.2: The upperbound on OBDD size for functions invariant under various transitive groups of degree 12 predicted by (4.3) for three ways of choosing orderings.

of (4.3) for each respective group and the ordering computed using the two variants

of the heuristic of the above section. The Random column is the mean upperbound computed from 10 runs on random orderings.

The table shows consistently better orderings with the Shortest heuristic and consistently worse than random with the Longest heuristic.

Conclusion This experiment is further illustration of how the structure of the known symmetry group of a problem can be used to optimise search in a way independent from the problem itself.

Related work Combining symmetry with OBDDs has been studied elsewhere, although the heuristic technique of this section seems not to have been looked at. A problem in model-checking of finite state systems in the presence of symmetry is that of encoding as an OBDD the equivalence relation on states induced by the symmetry. Upper bounds on the size of the OBDD for this relation for some groups was given in [CEFJ96]. Such an equivalence relation may have a high degree of symmetry itself, and so the approach of this section could be applied to this problem from the point of view of the symmetry of the function being encoded.

4.10 Summary

This chapter has set the context for this thesis in terms of the complexity questions that arise in practice in combining groups and propositional logic. We have seen a number of different ways of utilising symmetry within search, ways of finding symmetry, of manipulating the groups found and of the complexity problems associated with all these procedures. The next four chapters aim to get to the heart of some of these issues by tackling the complexity of group equivalence, containment and symmetry computation for formulas.

Chapter 5

Complexity Preliminaries

5.1 Introduction

This chapter collects some preparatory complexity results of relevance to the next three chapters. In §5.2 we review a previous result of Boy de la Tour concerning his MSYMM problem. We show that the problem is co-NPC with respect to polynomial transformation, improving his proof of completeness with respect to Turing reduction, a stronger notion of reducibility. We also look at the apparently more general GROUP SYMMETRY problem and show that it is also co-NPC with respect to polynomial transformation. Rather than obscure the Σ_2^p problems of the next chapters with membership proofs we include in this chapter in §5.3 a typical membership proof for these problems using G-EQUIV as an example. We discuss in §5.4 an example of where we can show that there is nothing special about the group W which makes complexity problems harder, even though it may make proofs easier using the Δ-Asg correspondence. The complexity of computing the closure construction of §3.2 is examined in §5.5 for various classes of formula. In §5.6 we review and fix some conventions.

5.2 MSYMM and GROUP SYMMETRY

MSYMM Boy de la Tour showed that the following problem was in co-NP and was NP-hard with respect to Turing reducibility [BdlT96a]. Given a wff $\phi(\vec{x})$ and a permutation $g \in W(\vec{x})$ the problem MSYMM asks whether g is a member of the semantic symmetry

group of ϕ

> Is g a semantic symmetry of ϕ? Formally is it the case that $\models \phi^g \leftrightarrow \phi$ is true?

His proof proceeds as follows. The complementary problem MSYMM^c is in NP because we can guess an assignment $a \in \mathsf{Asg}(\vec{x})$ in polynomial time and check that $\phi^g(a) \neq \phi(a)$ in polynomial time. Therefore MSYMM is in co-NP. Now we use MSYMM as an oracle to determine whether ϕ is invariant under each generator of $\Delta(\vec{x})$. We can then solve SAT in polynomial time as follows. We check one by one whether $\langle \phi, (x\ \neg x) \rangle$ is an instance of MSYMM for each $x \in \vec{x}$. If this fails then for some x there is an assignment $a \in \mathsf{Asg}(\vec{x})$ such that $\phi^{(x\ \neg x)}(a) = \phi(a^{(x\ \neg x)}) \neq \phi(a)$ and so ϕ is satisfiable and the answer is yes. If this is true for each x then we know that ϕ is either valid or false since it must be invariant under the group $\Delta(\vec{x})$ by Lemma 2.9.2, and this group acts transitively on the assignments $\mathsf{Asg}(\vec{x})$. So we just check an arbitrary assignment a to see if $\phi(a) = 1$. If so then ϕ is valid and so satisfiable and the answer is yes, otherwise no.

We use some routine constructions to show how MSYMM can be proved co-NPC by exhibiting polynomial transformations from and to the standard co-NPC problem UNSAT, the language of unsatisfiable boolean expressions.

Theorem 5.2.1 UNSAT $\equiv_P$ MSYMM and therefore MSYMM is co-NPC.

Proof MSYMM $\leq_P$ UNSAT since the validity of $\phi \leftrightarrow \phi^g$ is equivalent to the unsatisfiability of $\neg(\phi \leftrightarrow \phi^g)$. To show UNSAT $\leq_p$ MSYMM we must reduce any instance $\phi(\vec{x})$ of UNSAT to a symmetry problem. We utilise the gluing technique of Lemma 3.3.3 and the following propositional theorem:

$$\models (c \leftrightarrow \phi \vee c) \wedge (\neg c \leftrightarrow \phi \vee \neg c) \leftrightarrow \neg\phi.$$

We reduce by the following translation where c is any satisfiable, non-tautologous formula, say $\wedge\vec{x}$. This additional construct is necessary to accommodate the condition

in Lemma 3.3.3 that no component of the product expression is unsatisfiable.

$$\begin{aligned} \models \neg\phi \quad &\Leftrightarrow \quad \models c \leftrightarrow \phi \vee c \quad \& \quad \models \neg c \leftrightarrow \phi \vee \neg c && \text{(above identity)} \\ &\Leftrightarrow \quad \models \langle c, \neg c, \phi \vee c, \phi \vee \neg c\rangle^{h_3 h_2 h_4 h_2} \leftrightarrow \langle c, \neg c, \phi \vee c, \phi \vee \neg c\rangle. && \text{(see Example 3.3.4)} \end{aligned}$$

Thus we translate the unsatisfiability problem to one asking whether the constructed wff is invariant under the map $h_3h_2h_4h_2$. The assemblage of the map and product formula can clearly be performed in polynomial time and so the transformation is established.□

The basic idea of this proof is used again in Theorem 6.3.1 to show that the G-EQUIV and C-INVAR problems are polynomially equivalent. We now look at a problem which appears to more general than MSYMM but which we show to be polynomially equivalent to MSYMM.

The group symmetry problem Given a wff $\phi(\vec{x})$ and a group $G \leq W(\vec{x})$ the GROUP SYMMETRY problem asks

> Is every member of G a symmetry of ϕ? Formally, is it the case that $(\forall g \in G) \models \phi^g \leftrightarrow \phi$ is true?

Note that the group G in the problem instance is to be given in the form of a list of generators. This convention will apply to all the problems of this thesis. We show that this problem is co-NPC by showing that it is polynomially equivalent to MSYMM, shown above to be co-NPC.

Theorem 5.2.2 MSYMM $\equiv_P$ GROUP SYMMETRY and therefore GROUP SYMMETRY is co-NPC.

Proof MSYMM $\leq_p$ GROUP SYMMETRY by the following translation which sends a typical instance $\langle \phi, g\rangle$ of MSYMM to one of GROUP SYMMETRY involving the group $\langle g\rangle$ generated by the element g:

$$\models \phi^g \leftrightarrow \phi \quad \Leftrightarrow \quad (\forall h \in \langle g\rangle) \models \phi^h \leftrightarrow \phi.$$

$\Leftarrow$ is obvious since $g \in \langle g \rangle$. Conversely, if $\models \phi^g \leftrightarrow \phi$ then we know that $\models \phi^{g^m} \leftrightarrow \phi$ for all integers $m \geq 1$ by Lemma 2.9.1. To show GROUP SYMMETRY $\leq_p$ MSYMM we first recall from Lemma 2.9.2 that if $\models \phi^{g_k} \leftrightarrow \phi$ for each generator g_k of G then $\models \phi^g \leftrightarrow \phi$ holds for all $g \in G$. It therefore follows from the gluing technique Lemma 3.3.3 that

$$(\forall g \in G) \models \phi^g \leftrightarrow \phi \quad \Leftrightarrow \quad \models \langle \phi, \ldots, \phi \rangle^{\langle g_1, \ldots, g_i \rangle} \leftrightarrow \langle \phi, \ldots, \phi \rangle$$

where $g_1, \ldots, g_i$ are the generators for G. Note that $\Leftrightarrow$ holds in this case even if ϕ is a contradiction. We saw in Example 3.3.5 how to construct maps of form $\langle g_1, \ldots, g_i \rangle$. The reduction to an equivalent instance of MSYMM is clearly polynomial since the number i of generators of G is polynomial in the input size.□

5.3 Σ_2^p-membership for group quantified problems

In this section we give an example proof to show that the problem G-EQUIV is in the class Σ_2^p. Suitable encodings of wffs, assignments and permutations as binary strings are required. Assume $|\vec{x}| = n$. Recall from §2.3 that we can identify assignments $\mathsf{Asg}(\vec{x})$ with binary strings of length n. Permutations can be encoded as finite functions, that is lists of pairs $x \mapsto y$. Since we have n distinct identifiers, we would need $\Theta(\log n)$ bits to store a typical pair and thus $\Theta(n \log n)$ bits to store a permutation. Standard techniques can be applied to encode general structures such as boolean terms as binary strings of length polynomial in the size of the terms: see [GJ79] for details.

Theorem 5.3.1 The language G-EQUIV consisting of triples $\langle G, \phi, \psi \rangle$, encoding two wffs $\phi(\vec{x})$ and $\psi(\vec{x})$ and a group $G \leq W(\vec{x})$ given as a list of generator permutations, such that

$$(\exists g \in G) \models \phi^g \leftrightarrow \psi$$

is in the class Σ_2^p.

Proof As noted above we can encode assignments $\mathsf{Asg}(\vec{x})$ as strings of length n, and permutations as strings of length $cn \log n$ for some fixed constant c. Recall the mem-

bership criteria for Σ_2^p of Theorem 2.4.1. We show that the problem is of the form

$$(\exists g)(\forall a)\ R(\langle G, \phi, \psi\rangle, g, a)$$

where g, a are binary strings of bounded polynomial length as above representing group elements and assignments. We need to show that the relation R can be defined in terms of the problem and that it is polynomial time DTM recognisable. Define

$$R = \{\langle\langle G, \phi, \psi\rangle, g, a\rangle \mid g \in G \ \&\ (a \in \mathsf{Asg}(\vec{x}) \ \Rightarrow\ \phi(a^g) = \psi(a)\}.$$

So the relation R does the following: it checks that g is a well-formed permutation and that it is a member of G; then checks that a is a well-formed valuation and, if it is, checks that ϕ evaluated at the image of a under g is equal to ψ evaluated at a. We saw in Theorem 3.6.1 that membership testing for groups given as a list of generators is polynomial time. Constructing a^g is polynomial time. Evaluating a formula at some assignment is polynomial time. Therefore R is polynomial DTM recognisable. Now R checks membership for G *inside* the universal quantification for a, i.e. every time a new assignment is generated, but since g, G are not dependent upon a then this does not affect the equivalence of the statement to the instance

$$(\exists g \in G)(\forall a \in \mathsf{Asg}(\vec{x}))\ \phi^g(a) = \psi(a)$$

of G-EQUIV. Therefore G-EQUIV $\in \Sigma_2^p$.□

The above proof is prototypical for Σ_2^p membership of the problems of Chapters 6 and 7. Further membership claims will be informal, with references to this one.

5.4 Applying the D-transform

The correspondence we have noted in §2.8.1 between Δ and Asg may give rise to the impression that there is something peculiar about the group W (recall $\Delta < W$ is the base group of the wreath product W) that means complexity questions only become difficult when formulated using subgroups of $W(\vec{x})$ rather than the more intuitive symmetric group $S(\vec{x})$.

We show as an example how the G-EQUIV problem formulated in terms of subgroups of $W(\vec{x})$ can be reformulated in terms of subgroups of $S(\vec{x},\vec{x}')$ with $|\vec{x}| = |\vec{x}'|$ thus showing that there is nothing peculiarly hard about the $W(\vec{x})$ problems, nor is it strictly necessary to use this group in these cases except for ease of exposition. The D-transformation of §3.4 provides the appropriate tool. For G-EQUIV where $G \leq W(\vec{x})$ then we have

$$\begin{aligned}
(\exists g \in G) \models \phi^g \leftrightarrow \psi \quad &\Leftrightarrow \quad (\exists g \in G) \models D(\phi^g) \leftrightarrow D(\psi) && \text{(Lemma 3.4.1)} \\
&\Leftrightarrow \quad (\exists g \in G) \models D(\phi)^{D(g)} \leftrightarrow D(\psi) && \text{(Lemma 3.4.2)} \\
&\Leftrightarrow \quad (\exists d \in D(G)) \models D(\phi)^{D(g)} \leftrightarrow D(\psi)
\end{aligned}$$

with the last step being a consequence of the definition of $D(G)$ as $\{D(g) \mid g \in G\}$. We now have $D(G) < S(\vec{x},\vec{x}')$ as a subgroup of a larger symmetric group. Note that for instance $D(\Delta(\vec{x}))$ is simply a group isomorphic to $\Delta(\vec{x})$ over twice as many points and so we have not actually escaped from $\Delta(\vec{x})$, we have just reformulated it as a subgroup of a symmetric group such that the new G-EQUIV problem is equivalent to the original. This type of syntactic manipulation is a common feature of reductions in complexity theory (e.g. reducing CNF to 3-CNF to show that 3-CNF are equally hard for SAT) and in this case the manipulation is not inherently interesting. However we also make use of the D-transformation in Theorem 6.3.1 which invokes more interesting properties of it. Note that the MSYMM problem above need not be expressed in terms of elements of $W(\vec{x})$, but that $S(\vec{x})$ provides by this construction equally hard problems.

We shall refer to this section in future where similar arguments apply.

5.5 Complexity of computing $M_{k,\tau,\vec{x}}(\phi)$

The computation of $M_{k,\tau,\vec{x}}(\phi)$ (§3.2.1) can be realised with $O(n^k)$ *non-adaptive* tests of the form $\phi \models \bigvee s$ (for $\tau = C$) or $\bigwedge s \models \phi$ (for $\tau = D$). Non-adaptive means that the tests could be carried out in parallel, that is a test does not depend on the answers to previous tests. If we have a polynomial membership test for $M_{k,\tau,\vec{x}}(\phi)$ then this composes with the $O(n^k)$ number of tests to give an overall polynomial time procedure. Polynomial time satisfiability checking of 2-CNF or Horn formulas are

well-known results: see for example respectively [EIS76],[DG84]. If on the other hand the test is a co-NPC problem (e.g. if $\tau = C$ and ϕ is in k-CNF for $k \geq 3$) then a result of Buss and Hay [BH88] shows that $O(n^k)$ non-adaptive calls to a SAT oracle can be replaced with $O(\log n)$ adaptive calls. The technique uses a form of binary search to discover the exact number m of 'yes' answers to the $O(n^k)$ satisfiability problems followed by a further call of the form 'are m of these formulas satisfiable?' (a problem which is itself an NP problem) to extract the m certificates for the satisfiable formulas. Thus even if the basic test is co-NPC then computing $M_{k,\tau,\vec{x}}(\phi)$ is a problem in F-$P^{NP[\log n]}$. The following theorem summarises these considerations.

Theorem 5.5.1 Upper bounds for the complexity of computing $M_{k,\tau,\vec{x}}(\phi)$ are as follows:

i). Polynomial for ϕ in 2-CNF, 2-DNF, Horn, $\overline{\tau}$NF;

ii). No worse than F-$P^{NP[\log n]}$ for any other case.

Proof We examine the difficulty of individual tests of form $\phi \models \bigvee s$ or $\bigwedge s \models \phi$. The worst case analysis then follows from the above discussion.

i). Consider the case $\tau = C$. Then the test $\phi \models \bigvee s$ reduces to: satisfiability checking for 2-CNF or Horn when ϕ is in 2-CNF or Horn form, and is thus polynomial; a linear time problem when ϕ is in 2-DNF or DNF ($\overline{\text{C}}$NF), since we just check whether ϕ is satisfiable when all the s are false, a trivial problem for DNFs. The case for $\tau = D$ is dual: linear for 2-CNFs or Horn; polynomial for 2-DNFs or CNF ($\overline{\text{D}}$NF).

ii). UNSAT can be reduced to a test of form $\phi \models \bigvee\{\}$ therefore individual tests are co-NP-hard. The test $\phi \models \psi$ is clearly in co-NP, so even in the worst case the test can be solved by a SAT oracle.□

We look in §9.2.2 at a symmetry argument for simplifying this computation when the formula ϕ has a known symmetry group. We also discuss in Appendix B an implement-

ation for computing these representations which was used for some experimental work in Chapter 9. We exploit the tractability of computing closures of 3-Horn formulas to show how closures, or near closures, of 3-CNF formulas can sometimes be computed quite efficiently using a randomised technique.

5.6 Conventions applying to Chapters 6–8

We summarise some of our standard conventions for the next three chapters.

- The notation $\phi(\vec{x})$ means that $\mathsf{Vars}(\phi) \subseteq \vec{x}$. The sequence $\vec{x}$ is then used to induce finite types over $\mathsf{Lit}, W, \mathsf{Asg}, \mathsf{Cl}_k, \Delta$.
- Groups $G \leq W(\vec{x})$ are understood to be presented as finite lists of finite generators.
- Routine properties of group W which are not mentioned explicitly may be traced to §2.8.
- The taxonomy of Figure 5.1 shows the formula classes we consider in Chapters 6 and 7. An arrow $C_1 \to C_2$ indicates that any element of C_2 is also an element of C_1. We use implicitly many inference steps which say that a problem can be no harder for formula class C_2 than for C_1.

This chapter has introduced some of the main ingredients used in the following two chapters on equivalence and containment. In particular, we have established complexity results in the problem of applying the closure technique described in §3.2 for a number of different cases, which will be appealed to frequently in the remainder of this thesis.

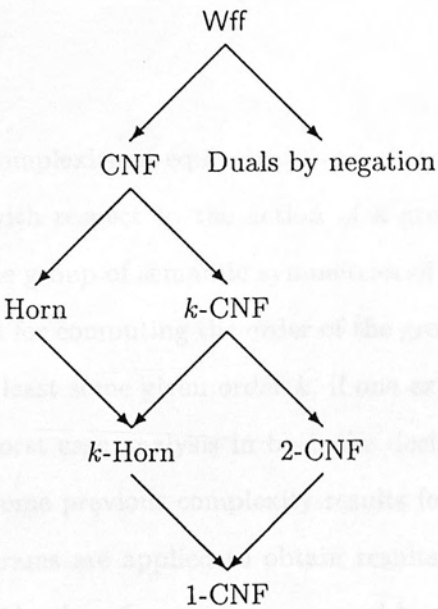


Figure 5.1: Taxonomy of formula classes

Chapter 6

Equivalence Problems

6.1 Introduction

This chapter examines the complexity of equivalence and associated invariance problems for boolean formulas with respect to the action of a group. We also examine the problem of computing the group of semantic symmetries of a formula. Worst case upper bounds are established for computing the order of the group, and for computing a group of symmetries of at least some given order k, if one exists. Particular classes of formula admit a better worst case analysis in both the decision problems and the symmetry group problem. Some previous complexity results for determining equivalence of binary decision diagrams are applied to obtain results for the complexity of determining G-equivalence of boolean functions represented by these structures. Tools from Chapter 3 will be applied as well as some of the preliminary complexity results of Chapter 5. The following chapter plan outlines the results in more detail. Issues arising from these problems will be discussed in Chapter 8.

6.1.1 Plan of chapter

In §6.2 we examine the problem (G-EQUIV) of determining whether two boolean formulas are logically equivalent under the action of a group G. We establish some lower bounds for the problem in terms of other known problems that can be reduced to it. The problem G-EQUIV is in the class Σ_2^p as was shown in §5.3 and we have not shown

it to be in any lower class in the general case. But for various classes of formula we can show that it exhibits improved behaviour. In particular if just one of the formulas is restricted to be in k-CNF for some fixed k then the problem can be shown to be in the class $\mathrm{P}^{\mathrm{NP}[\log n]}$. We also show that it is polynomially equivalent to OPS, a problem in NP, for (combinations of) k-Horn and 2-CNF.

In §6.3 and §6.4 we examine respectively the corresponding invariance problems C-INVAR and G-INVAR where we tackle the question of deciding whether a coset contains a semantic symmetry of the formula or whether a group contains a nontrivial semantic symmetry (i.e. not the identity). The former problem is shown to be polynomially equivalent to G-EQUIV. A Turing reduction of G-INVAR to C-INVAR illustrates how strong generator representations can be used to extract information about the semantic symmetry group: in this case G-INVAR is essentially the question of whether the order of the symmetry group is greater than 1. In §6.5 we show that the same technique can extract the order itself and we fix an upper bound for the complexity of this procedure. We also show that, given an integer k, the problem of determining whether ϕ has a symmetry group of order k is in Σ_2^p. For the k-τNF cases we show that for $k \geq 3$ then we can compute all symmetries in the class $\text{F-P}^{\mathrm{NP}[n^4]}$, or for 2-$\tau$NF or k-Horn, then the problem of computing the *semantic* symmetries is polynomially reducible to computing the automorphism group of a graph.

Illustrating a further example of where G-equivalence is relatively tractable is the case of a pair of binary decision diagrams with the groups acting on them as in §A.2. Here we show in §6.6 that undirected graphs can be encoded as binary decision diagrams such that the problem of G-equivalence for binary decision diagrams is at least as hard as GRAPH ISOMORPHISM. Previous results concerning testing equivalence of these diagrams give an easy result for an upper bound of NP in the general case for ordered binary decision diagrams and the class MA [Bab85] for the unordered and unreduced older versions, the 'free boolean graphs.'

6.2 The G-equivalence problem

In this section we study the complexity of the G-equivalence problem for boolean formulas (G-EQUIV): given wffs $\phi(\vec{x}), \psi(\vec{x})$ and a group $G \leq W(\vec{x})$ the question G-EQUIV asks

> Is ϕ logically equivalent to ψ under the action of the group G? Formally, is it the case that $(\exists g \in G) \models \phi^g \leftrightarrow \psi$ is true?

Note It is important to point out that the fact that the problem MSYMM is co-NPC by Theorem 5.2.1 (and so therefore is the simple generalisation to the problem $\models \phi^g \leftrightarrow \psi$) does not necessarily mean that adding an existential quantification ranging over a possibly exponential sized set leads to a Σ_2^p-complete problem. A number of pieces of evidence that G-EQUIV is *not* Σ_2^p-complete will be examined in §8.3.

6.2.1 Lower bounds

We show that a number of standard problems can be transformed to G-EQUIV thereby helping to delineate the lower boundary for the complexity of this problem. One interesting problem that we show G-EQUIV to be at least as hard as is the unique satisfiability problem studied in for instance [BG82]. The problem UNIQUESAT is, given wff ϕ:

> Is ϕ satisfied by a unique assignment to $\mathsf{Vars}(\phi)$?

The problem is in the class DP, but not known to be complete for this class with respect to polynomial transformation. It is suspected to lie outside NP $\cup$ co-NP. It has been shown that UNIQUESAT is NP-hard with respect to randomised polynomial transformation and that it is DP-complete with respect to randomised polynomial transformation. We summarise what we know about the lower boundary of G-EQUIV in the following theorem.

Theorem 6.2.1 The following reducibilities hold:

i). OPS $\leq_P$ G-EQUIV;

ii). UNSAT $\leq_P$ G-EQUIV;

iii). UNIQUESAT $\leq_P$ G-EQUIV;

iv). SAT $\leq_{RP}$ G-EQUIV;

v). SAT/UNSAT $\leq_{RP}$ G-EQUIV.

Proof

i). The construction of §3.5.2 showed how a set could be encoded as a simple formula. The following transforms a typical instance of OPS to an instance of G-EQUIV using Lemma 3.5.3

$$\begin{aligned}(\exists g \in G)\ s^g = t \quad &\Leftrightarrow \quad (\exists g \in G) \models \mathsf{Form}(s^g) \leftrightarrow \mathsf{Form}(t) \\ &\Leftrightarrow \quad (\exists g \in G) \models \mathsf{Form}(s)^g \leftrightarrow \mathsf{Form}(t).\end{aligned}$$

ii). A formula ϕ is unsatisfiable iff it is 1-equivalent to 0:

$$\models \neg\phi \quad \Leftrightarrow \quad (\exists g \in 1) \models 0^g \leftrightarrow \phi.$$

iii). Let ϕ be the minterm $\wedge\vec{y}$ where $\vec{y} = \mathsf{Vars}(\psi)$. Then

$$\psi \text{ has a unique model in } \mathsf{Asg}(\vec{y}) \quad \Leftrightarrow \quad (\exists \delta \in \Delta(\vec{y})) \models \phi^\delta \leftrightarrow \psi$$

since $\Delta(\vec{y})$ acts transitively on the set of minterms over $\vec{y}$ and ψ has a unique model iff it is equivalent to one of these minterms. Note that we could further apply the D-transform to this G-EQUIV instance, as shown in §5.4 to obtain an equivalent instance quantifying over a subgroup of a symmetric group.

iv). SAT $\leq_{RP}$ UNIQUESAT by [VV85] and UNIQUESAT $\leq_P$ G-EQUIV by iii).

v). SAT/UNSAT $\leq_{RP}$ UNIQUESAT by [VV85] and UNIQUESAT $\leq_P$ G-EQUIV by iii).□

Note that we have not been able to find a polynomial transformation from SAT to G-EQUIV. Since G-EQUIV appears the much harder of the two problems, this difficulty is interesting and is further discussed in §8.3.

Recall that we showed in §5.3 that G-EQUIV was in the class Σ_2^p. We do not know if it is Σ_2^p-complete and we examine why it might not be in §8.3. In the next section we look at special cases of G-EQUIV for certain classes of formula which show an improved behaviour in the sense of having a well-defined upper bound.

6.2.2 Breakdown

In Figure 6.1 we summarise what we know about the complexity of G-EQUIV for special classes of formula. Below we give a proof that in the restricted cases of G-EQUIV where both formulas must be in either 2-CNF or k-Horn then the problem is polynomially equivalent to the problem (OPS) of determining whether two sets are in the same orbit of a group. We also give a proof that the problem where just one of the formulas must be in k-CNF for fixed k is in the class $\mathrm{P}^{\mathrm{NP}[\log n]}$.

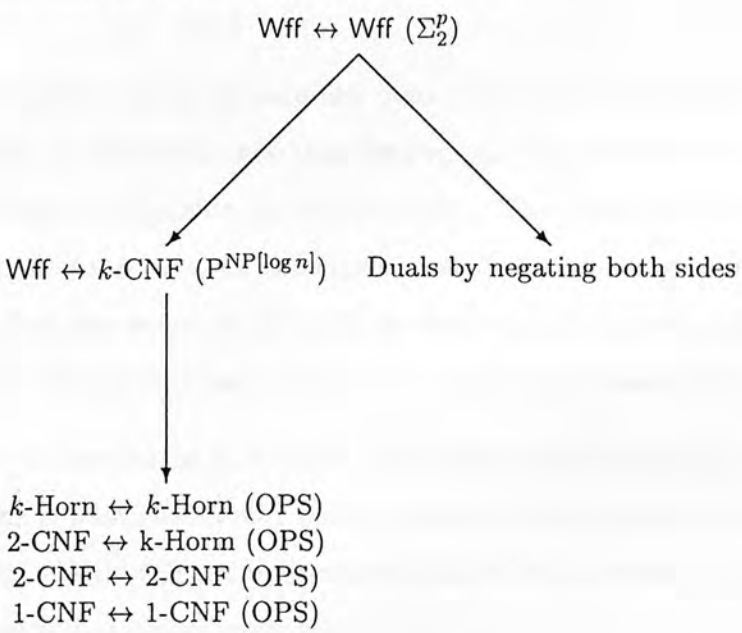


Figure 6.1: Upper bounds for G-EQUIV for pairs of formulas in the indicated classes.

Theorem 6.2.2 Consider the restriction R on G-EQUIV where $\phi(\vec{x}), \psi(\vec{x})$ must be in 2-CNF or k-Horn for some fixed $k \geq 1$. We show that G-EQUIV(R) is polynomially equivalent to OPS. If a further restriction is added to R so that the group G in any instance of the problem must be either $S(\vec{x})$ or $W(\vec{x})$ then G-EQUIV(R) is polynomially equivalent to GRAPH ISOMORPHISM.

Proof The reducibility OPS $\leq_P$ G-EQUIV(R) follows from part i) of Theorem 6.2.1 where the target formulas were in 1-CNF which is a subcase of both 2-CNF and k-Horn. Let j be the greater of 2 and k. We show G-EQUIV(R) $\leq_P$ OPS using the result of Theorem 5.5.1 for computing the maximal j-CNF representation of a wff. In the following we understand the subscripts on the operators M and J to be $j, C, \vec{x}$. Lemma 3.2.1 we have $J(\phi) \equiv \phi$ and $J(\psi) \equiv \psi$ and therefore by Lemma 3.2.3 $J(\phi^g) \equiv \phi^g$ for any $g \in W(\vec{x})$.

$$
\begin{aligned}
(\exists g \in G) \models \phi^g \leftrightarrow \psi \quad &\Leftrightarrow \quad (\exists g \in G)\ M(\phi^g) = M(\psi) && \text{(Lemma 3.2.2 twice)} \\
&\Leftrightarrow \quad (\exists g \in G)\ M(\phi)^g = M(\psi) && \text{(Lemma 3.2.3)} \\
&\Leftrightarrow \quad (\exists g \in G[\mathsf{Cl}_j(\vec{x})])\ M(\phi)^g = M(\psi) \\
&\in \quad OPS
\end{aligned}
$$

Recall that $G[\mathsf{Cl}_j(\vec{x})]$ is the group naturally induced on *sets* of literals by G acting on the literals $\mathsf{Lit}(\vec{x})$. It remains to show that the steps are polynomial. The computation of $M(\phi)$ and $M(\psi)$ is polynomial by Theorem 5.5.1. The group $G[\mathsf{Cl}_j(\vec{x})]$ has degree which is larger than that of G by a factor of just $O(n^j)$ (however it is of the same order). Recall that generators for $G[\mathsf{Cl}_j(\vec{x})]$ can be computed directly and easily from generators of G. We therefore have a polynomial time transformation into OPS.

Now consider the case where $G = S(\vec{x})$. We show G-EQUIV(R) $\leq_P$ GRAPH ISOMORPHISM. The opposite direction follows because we have already established that OPS reduces to G-EQUIV(R) and furthermore GRAPH ISOMORPHISM $\leq_P$ OPS by (see §2.5.1). We proceed as before, computing $M(\phi)$ and $M(\psi)$ then we use the graph construction of §3.5.1 to compute $\mathsf{Graph}_S(M(\phi))$ and $\mathsf{Graph}_S(M(\psi))$. Now we know by the properties of Lemma 3.5.1 that any map sending one graph into the other must be induced by some $s \in S(\vec{x})$ acting in a natural way on the vertices. In other words

the instance of GRAPH ISOMORPHISM

$$(\exists g \in S(\mathsf{Nodes}_j(\vec{x}))\ \mathsf{Graph}_S(M(\phi))^g = \mathsf{Graph}_S(M(\psi))$$

is true if and only if

$$(\exists g \in S(\vec{x})[\mathsf{Nodes}_j(\vec{x})])\ \mathsf{Graph}_S(M(\phi))^g = \mathsf{Graph}_S(M(\psi))$$

if and only if

$$(\exists g \in S(\vec{x}))\ M(\phi)^g = M(\psi)$$

and this is equivalent to our original instance of G-EQUIV for group $S(\vec{x})$ by the previous steps for an arbitrary G. Note that generators for $S(\mathsf{Nodes}_j(\vec{x}))$ can be constructed in polynomial time from generators for $S(\vec{x})$ and that the group has same order as $S(\vec{x})$. Finally using the extension to $W(\vec{x})$ of Lemma 3.5.1 we have equivalent results for the case that the group in the instance of the problem is $W(\vec{x})$. □

Theorem 6.2.3 Consider the restriction R on G-EQUIV where $\phi(\vec{x})$ must be in k-CNF for some fixed $k \geq 3$. We show that G-EQUIV(R) is in $\mathrm{P}^{\mathrm{NP}[\log n]}$.

Proof We first establish that the result is independent of the nature of the other formula $\psi(\vec{x})$ and this is because if ϕ, ψ are equivalent under some action of the group G then ψ must also be k-CNF representable. Let subscripts on J, M be understood as $k, C, \vec{x}$. We know $J(\phi) \equiv \phi$ by Lemma 3.2.1 and thus $J(\phi^g) \equiv \phi^g$ by Lemma 3.2.3. Now for any $g \in W(\vec{x})$

$$\begin{aligned}
\phi^g \equiv \psi &\Rightarrow M(\phi^g) = M(\psi) && \text{(Lemma 3.2.2 twice)}\\
&\Rightarrow M(\phi^g)^C \equiv M(\psi)^C && \text{(turn into k-CNF §3.2.1)}\\
&\Rightarrow J(\phi^g) \equiv J(\psi) && \text{(definition)}\\
&\Rightarrow \phi^g \equiv J(\psi) && \text{(since } J(\phi^g) \equiv \phi^g\text{)}\\
&\Rightarrow \psi \equiv J(\psi) && \text{(by assumption)}
\end{aligned}$$

We show that the problem can be solved by a polynomial time DTM using $O(\log n)$ calls to a SAT oracle where $n = |\vec{x}|$. So the algorithm begins as before by computing $M(\phi)$ and $M(\psi)$ which requires $O(\log n)$ SAT calls by Theorem 5.5.1. Next we make

one additional call to the SAT oracle of our Turing machine to test whether $M(\phi)^C \stackrel{\text{def}}{=} J(\psi) \equiv \psi$. If this fails then ψ is not k-CNF representable and so, by the above result, cannot be equivalent to any image of ϕ (which is in k-CNF). The algorithm would then reject the instance of G-EQUIV(R). If the test succeeds then we proceed to transform the problem to an equivalent instance of OPS by the argument used in the proof of Theorem 6.2.2. Now OPS is in NP and so it requires just one more call to the SAT oracle to decide the instance of OPS and hence of the original G-EQUIV problem. Therefore we have shown that the decision problem for G-EQUIV(R) is in $\mathrm{P}^{\mathrm{NP}[\log n]}$ since the extra two calls do not affect the overall $O(\log n)$ result.□

In §8.3 we add some comments to the general question of whether SAT can be polynomially transformed to G-EQUIV on the question as to whether the restricted k-CNF case can be complete for $\mathrm{P}^{\mathrm{NP}[\log n]}$.

6.3 The C-invariance problem

In this section we study the C-invariance problem for boolean formulas (C-INVAR). Given wff $\phi(\vec{x})$ and coset $C = Gk$ with $G \leq W(\vec{x})$ and $k \in W(\vec{x})$ the C-INVAR problem asks:

> Does the coset C contain a semantic symmetry of ϕ? Formally, is it the case that $(\exists g \in C) \models \phi^g \leftrightarrow \phi$ is true?

Note that C does not contain the identity group element unless $k \in G$ so the problem is not trivial in general. We show by the following that the problems G-EQUIV and C-INVAR are polynomially equivalent.

Theorem 6.3.1 C-INVAR $\equiv_p$ G-EQUIV.

Proof C-INVAR $\leq_p$ G-EQUIV is established by the following:

$$\begin{aligned}(\exists g \in Gk) \models \phi^g \leftrightarrow \phi \quad &\Leftrightarrow \quad (\exists j \in G) \models \phi^{jk} \leftrightarrow \phi \\ &\Leftrightarrow \quad (\exists j \in G) \models \phi^j \leftrightarrow \phi^{k^{-1}}\end{aligned}$$

which is now clearly an instance of G-EQUIV with $\psi = \phi^{k^{-1}}$. For G-EQUIV $\leq_p$ C-INVAR, informally we want to transform a typical instance of G-EQUIV using the gluing technique of §3.3 to the question of whether

$$\models \langle \phi, \psi \rangle^{gh^{-1}j} \leftrightarrow \langle \phi, \psi \rangle$$

for some $g, j \in G$ where h is the renaming permutation, since Lemma 3.3.2 then seems to provide $\models \phi^g \leftrightarrow \psi$ and $\models \psi^j \leftrightarrow \phi$. Note that $gh^{-1}j$ is an element of the right coset $G(h^{-1}Gh)h^{-1} = Gh^{-1}G$. This lemma only works however given that neither of ϕ, ψ is a contradiction, provision for which does not appear to be easily incorporated into the transformation. Determining in advance whether they are contradictions would not constitute a provably polynomial transformation. We need to construct the transformation using formulas which are certainly not contradictions, as in the proof of Theorem 5.2.1. We employ a similar technique. However in this case we would need a satisfiable, non-tautologous formula c which is *invariant* under the group G. We observed in §3.4 that this can be problematic for some subgroups of $W(\vec{x})$.

We start by applying the D-transformation to a typical instance of G-EQUIV obtaining the question $(\exists d \in D(G)) \models D(\phi)^d \leftrightarrow D(\psi)$ following the argument of §5.4. We can now transform this to an equivalent instance of C-INVAR by the technique of adding the formula $c \stackrel{\text{def}}{=} \wedge \vec{x}, \vec{x}'$ (the $\vec{x}'$ are the variables added by D) to both $D(\phi)$ and $D(\psi)$, ensuring that the two glued components are satisfiable:

$$(\exists \gamma \in D(G)h^{-1}D(G)) \models \langle D(\phi) \vee c, D(\psi) \vee c \rangle^\gamma \leftrightarrow \langle D(\phi) \vee c, D(\psi) \vee c \rangle$$

This instance is equivalent to the original instance of G-EQUIV as follows. Using Lemma 3.3.2 we have that the above is true if and only if $\models (D(\phi) \vee c)^{d_1} \leftrightarrow (D(\psi) \vee c)$ and $\models (D(\psi) \vee c)^{d_2} \leftrightarrow (D(\phi) \vee c)$ for some $d_1, d_2 \in D(G)$. Using the facts that c is invariant under $D(G)$ ($D(G)$ is a subgroup of $S(\vec{x}, \vec{x}')$ so $c^{d_1} \equiv c$ and $c^{d_2} \equiv c$) and that both $c \not\models D(\phi)$ and $c \not\models D(\psi)$ (since e.g. $D(\phi)$ is only true when the $\vec{x}'$ take different values to the $\vec{x}$), we obtain $\models D(\phi)^{d_1} \leftrightarrow D(\psi)$ and $\models D(\psi)^{d_2} \leftrightarrow D(\phi)$ for some $d_1, d_2 \in D(G)$. Putting $d_1 = D(g)$ and $d_2 = D(j)$ and applying Lemma 3.4.2 we obtain that $\models \phi^g \leftrightarrow \psi$ and $\models \psi^j \leftrightarrow \phi$, for some $g, j \in G$. This is now equivalent to

our original instance $(\exists g \in G) \models \phi^g \leftrightarrow \psi$ of G-EQUIV because if this holds for g' then put $g = g'$ and $j = g'^{-1}$ and if the former holds then the latter is immediate.□

6.4 The G-invariance problem

In this section we study the G-invariance problem for boolean formulas (G-INVAR). Given wff $\phi(\vec{x})$ and group $G \leq W(\vec{x})$ the G-INVAR problem asks:

> Does the group G contain a nontrivial semantic symmetry of ϕ? Formally, is it the case that $(\exists g \in G \setminus 1) \models \phi^g \leftrightarrow \phi$ is true?

Theorem 6.4.1 G-INVAR $\leq_T$ C-INVAR.

Proof We use an oracle for C-INVAR to solve G-INVAR in polynomial time. We refer to the material of §3.6 in order to show that $G \setminus 1$ can be expressed as a union of a polynomial number of right cosets of subgroups of G. The cosets in question are as follows: starting with C_0, the transversal $G_0 : G_1$, we take the union of all cosets except the identity coset. These add up to $G_0 \setminus G_1$. Similarly we construct $G_1 \setminus G_2$ so that by taking the union we obtain $(G_0 \setminus G_1) \cup (G_1 \setminus G_2) = G_0 \setminus G_2$ and so on until we have $G_0 \setminus 1 = G \setminus 1$. Thus we reduce an instance of G-INVAR to $O(n^2)$ calls to C-INVAR, returning true overall if any coset contains a symmetry of ϕ, false otherwise.

The G-INVAR problem would seem to be the more natural symmetry problem compared with C-INVAR. However the structure of the problem seems to prevent any easy demonstration of polynomial equivalence with C-INVAR and G-EQUIV. The above proof will be used in a more general sense later to show how we can compute the total *number* of symmetries using the same amount of work.

6.5 Computing semantic symmetries

Boy de la Tour showed that his problem GSYMM of computing a generating set for the group of semantic symmetries $\Sigma(\phi)$ of ϕ was NP-hard by illustrating a Turing

reduction from MSYMM to GSYMM: given an oracle for computing $\Sigma(\phi)$ we can decide an instance $\langle \phi, g \rangle$ of MSYMM by testing $g \in \langle A \rangle$ where A is the list of generators returned by the orcacle. This composes with his result of NP-hardness for MSYMM to give an NP-hardness result for GSYMM.

A full analysis of the complexity of GSYMM is beyond the author but we add some results on upper bounds and give results for special cases. In the general case we can give upper bounds for these two problems: compute the order of $\Sigma(\phi)$; output a group of semantic symmetries of order k if ϕ has one, otherwise output 'no'? For special cases where $\phi(\vec{x})$ is in k-τNF then we can compute $\Sigma(\phi)$ by computing $M_{k,\tau,\vec{x}}(\phi)$ and then using Lemma 3.2.3 to reduce the problem to computing the setwise stabiliser of the clause system. Since the group we are working from is $W(\vec{x})$ then we can utilise a further construction to reduce the problem to computing generators for the automorphism group of a graph Lemma 3.5.1.

6.5.1 Computing order of $\Sigma(\phi)$

The above reduction of Theorem 6.4.1 illustrates how one piece of information about $\Sigma(\phi)$ can be extracted, namely the order of the group. Let G be $W(\vec{x})$ in this case and assume we have constructed a strong generating set for G in polynomial time. In checking each $G_i c$ with $c \in C_i$ for a symmetry of ϕ we are actually doing a lot of the work in building a strong generating set for $\Sigma(\phi)$. If $\Sigma(\phi)$ contains an element which fixes $x_1, \ldots, x_i$ and sends x_{i+1} to x_{i+1}^c then it must be in the coset $G_i c$. Hence we know that a strong generating set for $\Sigma(\phi)$, based on the same ordering, consists of *some* element from each coset $G_i c$ for each $0 \leq i \leq n-2$ and $c \in C_i$ such that C-INVAR returns true. So although we don't know which element applies from each coset containing a symmetry, we know the number of elements in each transversal $\Sigma(\phi)_i : \Sigma(\phi)_{i+1}$. The order of $\Sigma(\phi)$ is given by multiplying together the sizes of these transversals. Now we need $O(n^2)$ calls to the C-INVAR oracle to extract this information and C-INVAR is in Σ_2^p and we do not know that it is in any lower class. Hence our best upper bound for computing the order of the semantic symmetry group is the class F-Δ_3^p (recall that

$\Delta_3^p \stackrel{\text{def}}{=} \mathrm{P}^{\Sigma_2^p}$) where we make no more than $O(n^2)$ (nonadaptive) calls to the Σ_2^p oracle, outputting the order after an easy computation on the sizes of the transversals.

6.5.2 Find a symmetry group of order k

Given k the question of whether $\phi(\vec{x})$ is logically invariant under some subgroup G of $W(\vec{x})$ of order k is a problem in $\Sigma_2^p \stackrel{\text{def}}{=} \mathrm{NP}^{\mathrm{NP}}$. We can guess a set of generators of G in polynomial time (we saw in §3.6 that every group of degree n has a generating set with $O(n^2)$ elements) verify that $|G| = k$ in polynomial time (Theorem 3.6.1) and verify that G in in fact a symmetry group of ϕ in one call to an NP oracle, by Theorem 5.2.2 which showed that this problem is co-NPC. The problem of outputting the generators is therefore in the function class F-Σ_2^p.

Note It would seem simple to combine these two upper bounds to give an upper bound for computing $\Sigma(\phi)$ from scratch. However there is technical problem here, unresolved by the author, of combining deterministic and nondeterministic function classes: it is not clear that inclusion of the underlying decision classes is trivially preserved under the F annotation.

6.5.3 Special cases

Using the closure construction of §3.2 we can extract semantic symmetries of k-CNF and k-DNF formulas by applying standard automorphism analysis to the closed clause systems using Lemma 3.2.3. The efficiency of this is of course dependent upon the efficiency of computing the closure and we summarise in the following theorem:

Theorem 6.5.1 For fixed k, then computing $\Sigma(\phi)$ for ϕ in k-τNF is

i). polynomially equivalent to the problem of finding generators for the automorphism group of a graph, for $k = 2$ or if ϕ is in k-Horn;

ii). in F-$\mathrm{P}^{\mathrm{NP}[n^4]}$ for $k \geq 3$.

Proof

i). By Theorem 5.5.1 and Lemma 3.2.1 we have a polynomial time algorithm for computing $M_{k,\tau,\vec{x}}(\phi)$ and then by Lemma 3.2.3 we have that the semantic symmetries correspond to the stabiliser of $M_{k,\tau,\vec{x}}(\phi)$ in $W(\vec{x})$. However we can reduce this to computing graph automorphisms by Lemma 3.5.1 since our parent group is $W(\vec{x})$. For the other direction, we can use Method 2 of §3.5.3 to encode a graph as a 1-CNF which generalises to both 2-CNF and k-Horn.

ii). We have a computation of $M_{k,\tau,X}(\phi)$ in polynomial time using $O(\log n)$ calls to a SAT oracle by Theorem 5.5.1. Again we can now reduce this to computing graph automorphisms, which can be performed using $O(n^4)$ adaptive calls to a GRAPH ISOMORPHISM oracle (see [Hof82]). Now GRAPH ISOMORPHISM is in NP and so we can definitely do the whole job using $O(n^4 + \log n) = O(n^4)$ SAT calls. Therefore the overall problem is in F-$\mathrm{P}^{\mathrm{NP}[n^4]}$ in the worst case.

6.6 Boolean graphs

Some results from the literature can be applied to the problem of testing G-equivalence for a pair of 'free boolean graphs'. These were precursors to the now familiar binary decision diagrams Appendix A. They are basically BDDs with no reduced or ordering constraints but requiring that complementary literals do not appear on any path from the root to a terminal. In [FHS78] it was shown that if just one of a pair of boolean graphs was ordered in the same sense as i) of the definintion of OBDD (§A.2) then equivalence could be tested in polynomial time by reducing the problem to testing equivalence of two deterministic finite automata. In [BC80] it was later shown that one could relax the condition of one of the graphs being ordered at the cost of making do with a randomised polynomial time algorithm for the equivalence check.

Consider Method 1 for constructing boolean formulas from graphs §3.5.3. We can easily proceed to encode such formulas as OBDDs since they consist merely of a small list of minterms, and the number of paths in an OBDD cannot exceed the number of models. Hence we have

Theorem 6.6.1 G-EQUIV is harder than GRAPH ISOMORPHISM for a pair of OBDDs (and hence boolean graphs) and is

i). in NP for a pair of OBDDs

ii). in MA (see note below) for a pair of free boolean graphs.

Proof Recall that a test $g \in G$ with G given as a list of generators can be performed in polynomial time (Theorem 3.6.1). From the above discussion we see that the graph isomorphism problem can be reduced to the G-EQUIV problem for OBBDs by encoding the two graphs as OBDDs.

i). The first case is in NP since we can guess a group element $g \in G$ and apply it to ϕ according to §A.2. Then we use the algorithm of [FHS78] to check in polynomial time whether the graphs of ϕ^g and ψ define the same function.

ii). The problem fits the pattern of a 'guess and check' protocol with the checking part performed by Arthur in randomised polynomial time. If the two graphs are equivalent under the permutation g provided by Merlin then the algorithm of [BC80] applied to ϕ^g and ψ will accept with probability at least $1/2$ (g is applied to the graph in the obvious way by its action on vertex labels, as for BDDs). Using 'amplification', i.e. running the randomised test more than once, the error bound can be decreased to less than $1/3$ as required by the definition of MA.

Note The class MA introduced in [Bab85] (we look at some interactive proof classes in §8.3) has also been called 'nondeterministic BPP' [Joh88], fitting exactly the format of testing G-equivalence for boolean graphs described here.

6.7 Summary

We have presented three equivalence problems in the class Σ_2^p and shown lower bounds that indicate that they are hard problems. We have established better upper bounds than Σ_2^p for special cases. We have also included some remarks on the complexity

of similar questions concerning boolean graphs. We have analysed the question of computing groups of semantic symmetries in terms of the equivalence problems and have shown upper bounds for computing the order of the group and for computing a group of order k if there is one. Some of the more interesting unresolved cases of this chapter are summarised in §8.7. In Chapter 8 we return to discussion of the equivalence problems in conjunction with results for the containment problems of the next chapter. In particular the question of whether the problem G-EQUIV is Σ_2^p-complete will be addressed.

Chapter 7

Containment Problems

7.1 Introduction

This chapter examines the complexity of containment problems for boolean formulas with respect to the action of a group. In contrast to the decision problems of the previous chapter these problems are shown to be complete for the class Σ_2^p. Tools from Chapter 3 will be applied as well as some of the preliminary complexity results of Chapter 5. The following chapter plan outlines the results in more detail. Issues arising from these problems will be discussed in the next chapter.

7.1.1 Plan of chapter

In 7.2 we examine the problem (G-EMBED) of determining whether one boolean formula is logically contained in another under the action of a group G: in other words whether there is an element g of G such that $\models \phi^g \rightarrow \psi$. We establish straightforwardly that the problem is Σ_2^p-complete and remains so even if the groups are restricted to be subgroups of the symmetric group on the variables of the formulas. For various classes of formula we show improved worst case analysis. In particular if the formula ψ is restricted to be in k-CNF for some fixed k then we show that the decision problem is in the class $\mathrm{P}^{\mathrm{NP}[\log n]}$. This class is in Δ_2^p which is believed to be contained strictly in Σ_2^p. In the previous chapter we established a similar result for G-EQUIV. However, not being able to establish that it was Σ_2^p-complete, we could not say that the $\mathrm{P}^{\mathrm{NP}[\log n]}$

result for k-CNFs was definitely better than the worst possible case in the context of current understanding that the latter class is strictly contained in Σ_2^p. In the G-EMBED case it is therefore 'very probable' that k-CNFs are easier for this problem. We also analyse a number of other combinations of formula type giving more tractable cases.

We examine an apparently different but closely related problem (G-ISF) in §7.3. This problem is to determine whether a formula ϕ 'contains' another satisfiable formula which is invariant under a given group (unsatisfiable formulas are contained in this sense in any formula and are always invariant under any group, hence the condition on being satisfiable). In other words whether there is a satisfiable ψ such that $\models \psi \rightarrow \phi$ and ψ is invariant under G in the sense of the problem GROUP SYMMETRY of §5.2. In fact, although this is an intuitive way of understanding the problem and relates it in a nice way to graph theoretic questions of §8.4, it is easier to express it in another way. Namely, is it true that the conjunction of images of ϕ over elements of G is satisfiable? We show that this problem is Σ_2^p-complete using a kind of dual process to that which established the same result for G-EMBED. We also establish some more tractable cases for the problem for other classes of formula.

In the G-ISF problem we can quantify existentially over a *family* of groups without affecting the worst case complexity of the problem. That is we can ask if there is a $G \in \mathcal{F}$ such that $\langle G, \phi \rangle$ is an instance of G-ISF and we show in §7.4 that this problem is also Σ_2^p-complete. (We can also do this with G-EMBED but because of the nature of the question it would be less interesting and would in fact just be quantifying over the group $\langle \mathcal{F} \rangle$.) The family question ($\mathcal{F}$-ISF) in fact appears to be more natural than G-ISF and we examine some connections with graph-theoretic problems in Chapter 8.

A final problem, examined in 7.5, is the problem (C-DISJ) of determining whether a coset C contains an element g such that the disjunction of ϕ and ϕ^g is valid. This problem is proved Σ_2^p-complete by showing that it is polynomially equivalent to G-EMBED. The equivalence proof mirrors in many senses the equivalence proof for the problems G-EQUIV and C-INVAR of the previous chapter. The problem C-DISJ is interesting in that it can be viewed as a containment problem similar to G-EQUIV

involving only one formula, therefore answering negatively the question of whether it is the two formulas of G-EMBED that give that containment problem its hardness.

7.2 The G-embedding problem

In this section we examine the complexity of the G-embedding, or G-containment, problem for boolean formulas (G-EMBED): given wffs $\phi(\vec{x}), \psi(\vec{x})$ and a group $G \leq W(\vec{x})$ the question G-EMBED asks

> Is ϕ contained in ψ under the action of the group G? Formally, is it the case that $(\exists g \in G) \models \phi^g \rightarrow \psi$ is true?

7.2.1 Breakdown

In Figure 7.1 we summarise what we know about the complexity of G-EMBED for special classes of formula. Below we give a proof that the 1-CNF $\rightarrow$ 3-DNF case, that is where ϕ is in 1-CNF and ψ in 3-DNF, is complete for the class Σ_2^p, as therefore is any generalisation (see Figure 5.1). Since a 1-CNF formula is also in DNF then we have Σ_2^p-complete generalisations DNF $\rightarrow$ DNF and, by contraposition, CNF $\rightarrow$ CNF. We show that the 1-CNF $\rightarrow$ 1-CNF case is NPC. Any generalisation of this case which also admits a polynomial time check for $\models \phi^g \rightarrow \psi$ for given g is therefore also NPC. This is certainly the case when ϕ is in Horn form or 2-CNF and these cases are therefore NPC. We show that if ψ is restricted to be in k-CNF for any fixed k then the problem G-EMBED is in the class $\mathrm{P}^{\mathrm{NP}[\log n]}$. The j-CNF $\rightarrow$ k-CNF case is indicated in Figure 7.1 with the same worst case analysis. We do not know if there is a better analysis of this case for $j \geq 3$ than when ϕ may be any arbitrary formula. The 1-CNF $\rightarrow$ 2-DNF case is in NP since the test $\models \phi^g \rightarrow \psi$ is polynomial by virtue of the efficiency of checking 2-DNFs for validity. However we have not shown the G-EMBED problem to be either polynomial or NPC in this case. The 1-CNF $\rightarrow$ 1-DNF is an example of a degenerate case which can be solved easily be checking to see if one of the orbit of the literals of ϕ under G is a literal in ψ.

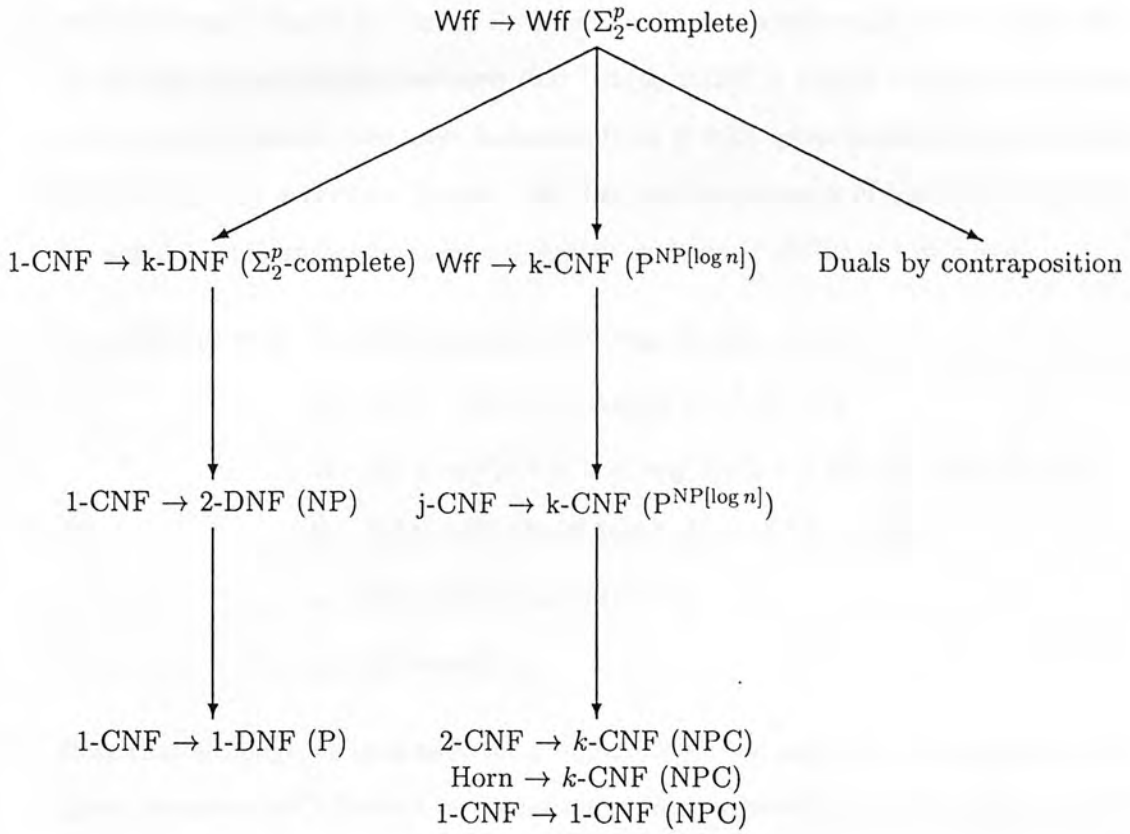


Figure 7.1: Upper bounds for the problem G-EMBED for pairs of formulas in the indicated classes.

There are corresponding dual questions obtained by contraposition. For instance the Wff → k-CNF case and the k-DNF → Wff are dual and will have polynomially equivalent complexity. All we need to do to transform one into the other is to apply negation to both sides, swap them round and apply de Morgan's laws to obtain a DNF from a CNF and vice versa.

7.2.2 Proofs

Theorem 7.2.1 G-EMBED is Σ_2^p-complete.

Proof It is clear from the proof of Theorem 5.3.1 that G-EMBED $\in \Sigma_2^p$ since we just replace the test $\phi^g(a) = \psi(a)$ with $\phi^g(a) \leq \psi(a)$. The transformation from the standard complete problem B_2 with instance $\phi(\vec{x}, \vec{y})$ for Σ_2^p of Theorem 2.4.2 can be done as follows. Note that we can use $\langle a, b \rangle$ with $a \in \mathsf{Asg}(\vec{x})$ and $b \in \mathsf{Asg}(\vec{y})$ to range

over elements of $\mathsf{Asg}(\vec{x}, \vec{y})$. Let $a_0 \in \mathsf{Asg}(\vec{x})$ be the zero assignment $x \mapsto 0$. Then using the Δ-Asg correspondence we know that $\mathsf{Orb}(a_0, \Delta(\vec{x})) = \mathsf{Asg}(\vec{x})$ and so the first step is to replace quantification over assignments to $\vec{x}$ with quantification over the group $\Delta(\vec{x})$ acting on the zero assignment. We then use the property of the formula $\bigwedge\vec{x}$ that for any $\delta \in \Delta(\vec{x})$ and assignment $a \in \mathsf{Asg}(\vec{x})$ we have $(\bigwedge\vec{x})^\delta(a) = 1 \Leftrightarrow a = a_0^\delta$.

$$
\begin{aligned}
(\exists\vec{x})(\forall\vec{y}) \models \phi \quad &\Leftrightarrow \quad (\exists a \in \mathsf{Asg}(\vec{x}))(\forall b \in \mathsf{Asg}(\vec{y}))\ \phi(\langle a, b\rangle) = 1 \\
&\Leftrightarrow \quad (\exists \delta \in \Delta(\vec{x}))(\forall b \in \mathsf{Asg}(\vec{y})\ \phi(\langle a_0^\delta, b\rangle) = 1 \\
&\Leftrightarrow \quad (\exists \delta \in \Delta(\vec{x}))(\forall \langle a, b\rangle \in \mathsf{Asg}(\vec{x}, \vec{y}))\ a = a_0^\delta \ \Rightarrow \ \phi(\langle a, b\rangle) = 1 \\
&\Leftrightarrow \quad (\exists \delta \in \Delta(\vec{x}))(\forall c \in \mathsf{Asg}(\vec{x}, \vec{y}))\ (\bigwedge\vec{x})^\delta(c) \leq \phi(c) \\
&\Leftrightarrow \quad (\exists \delta \in \Delta(\vec{x})) \models (\bigwedge\vec{x})^\delta \to \phi \\
&\in \quad \text{G-EMBED}
\end{aligned}
$$

Note that although we have used the group $\Delta(\vec{x})$ we can apply the D-transform to the above instance of G-EMBED to obtain an instance quantifying over a subgroup of a symmetric group. The details are the same as in §5.4 where $\leftrightarrow$ can be replaced with $\to$ by Lemma 3.4.1.□

Note that in the target instance of G-EMBED above then the left formula is in 1-CNF and the right formula is the original formula of the B_2 problem for which 3-DNF is sufficient to give Σ_2^p-completeness. Since the D-transform adds a 2-CNF formula to each formula then the D-transformed problem is not a 1-CNF $\to$ 3-DNF case even though, giving equivalent instances, the general problem is still Σ_2^p-complete.

Technically a 1-CNF formula is also a DNF formula containing just one clause. Therefore the 1-CNF $\to$ 3-DNF case generalises to DNF $\to$ 3-DNF which generalises to DNF $\to$ DNF. Applying contraposition we obtain the CNF $\to$ CNF case, which is therefore Σ_2^p-complete. Applying the D-transform to a CNF $\to$ CNF instance adds a 2-CNF formula and so the CNF $\to$ CNF case is Σ_2^p-complete for subgroups of the symmetric group on variables. This gives a nice uniform Σ_2^p-complete case:

Lemma 7.2.2 Let R be the restriction on G-EMBED such that both formulas must

be in CNF and G must be a subgroup of the symmetric group on formula variables. Then G-EQUIV(R) is Σ_2^p-complete.

We now show that even when ϕ, ψ are both in 1-CNF, the G-EMBED problem is NPC.

Theorem 7.2.3 Let R be the restriction applied to G-EMBED where both ϕ and ψ must be in 1-CNF. Then G-EMBED(R) is NPC.

Proof Given $g \in W(\vec{x})$ then the test $\models \phi^g \rightarrow \psi$ is easily seen to be polynomial, therefore G-EMBED(R) is certainly in NP. We show NP-completeness by reduction from SUBGRAPH ISOMORPHISM. Let the pair of directed graphs $H_1 = \langle V, E_1 \rangle$, $H_2 = \langle V, E_2 \rangle$ be a typical instance of SUBGRAPH ISOMORPHISM. Then using Method 2 of §3.5.3 to encode graphs as formulas:

$$\begin{aligned} (\exists g \in S(V)) E_1^g \subseteq E_2 &\Leftrightarrow (\exists g \in S(V)) \models \mathsf{Form}(E_1^g) \rightarrow \mathsf{Form}(E_2) \qquad \text{(Lemma 3.5.3)} \\ &\Leftrightarrow (\exists g \in S(V)[V^2]) \models \mathsf{Form}(E_1)^g \rightarrow \mathsf{Form}(E_2) \\ &\in \text{G-EMBED}(R) \end{aligned}$$

where the group $S(V)[V^2] \leq S(V^2)$ is the group induced naturally on the edges V^2 by $S(V)$ acting pointwise on the vertices V. Note that it is easy to construct generators for $S(V)[V^2]$. One simply specifies how each generator of $S(V)$ acts on the set V^2.□

If the test $\models \phi^g \rightarrow \psi$ is polynomial for any g then G-EMBED in this case is in NP. Furthermore if it is a generalisation of the above 1-CNF case then the restriction must be NPC. We summarise this in the following theorem.

Theorem 7.2.4 Let R be the restriction on G-EMBED where ϕ is in 2-CNF or Horn, and ψ is in k-CNF for fixed $k \geq 1$. Then G-EMBED(R) is NPC.

Proof G-EMBED(R) is clearly NP-hard with respect to polynomial transformation since it is a generalisation of the 1-CNF $\rightarrow$ 1-CNF case. We need to show it is in NP. Given any $g \in W(\vec{x})$ then the test $\models \phi^g \rightarrow \psi$ is in fact polynomial. In the proof of Theorem 5.5.1 we noted that a test of form $\phi \models \bigvee s$ is polynomial for ϕ in 2-CNF or Horn. To check entailment of a CNF just requires checking entailment for each of the

(polynomial number of) clauses. Note that if ϕ is in Horn then ϕ^g may not be since g may alter the polarity of literals. However we have

$$\phi^g \models \psi \quad \Leftrightarrow \quad \phi \models \psi^{g^{-1}}$$

so we can just apply the inverse of the permutation to the k-CNF formula. Therefore G-EMBED(R) is NPC.□

We now apply the $M_{k,\tau,\vec{x}}$ closure technique of §3.2 to establish apparently better than worse case (Σ_2^p-complete) analysis for the special case where ψ, the formula on the right in G-EMBED instances, is in k-CNF for some fixed k. The formula on the left can be anything.

Theorem 7.2.5 Let R be the restriction on G-EMBED where ψ is in k-CNF for some fixed $k \geq 3$. Then G-EMBED(R) is in $\mathrm{P}^{\mathrm{NP}[\log n]}$.

Proof The proof is a slightly simpler version of the proof of the corresponding Theorem 6.2.3 for G-EQUIV. This time we do not need to establish that ϕ is also k-CNF representable. It is sufficient to show that the largest k-CNF entailed by ϕ is embeddable in ψ. Let subscripts of M, J be understood as $k, C, \vec{x}$. Then

$$\begin{aligned}
(\exists g \in G) \models \phi^g \rightarrow \psi \quad &\Leftrightarrow \quad (\exists g \in G) M(\psi) \subseteq M(\phi^g) \quad \text{(Lemma 3.2.2)} \\
&\Leftrightarrow \quad (\exists g \in G) M(\psi) \subseteq M(\phi)^g \quad \text{(Lemma 3.2.3)} \\
&\in \quad \mathrm{NP}
\end{aligned}$$

We therefore have a polynomial time transformation of G-EMBED(R) using $O(\log n)$ SAT calls by Theorem 5.5.1 to an NP problem, requiring just one more SAT call, so the decision problem for G-EMBED(R) is in $\mathrm{P}^{\mathrm{NP}[\log n]}$.□

The fact that G-embedding appears strictly easier for k-CNF that for CNF (despite well-known equivalent hardness for these formulas in a satisfiability context) is one of the more interesting points raised by the thesis and will be discussed in §8.2.

7.3 The G-invariant subformula problem

In this section we examine the G-invariant subformula problem for a boolean formula (G-ISF): given wff $\phi(\vec{x})$ and group $G \leq W(\vec{x})$ the question G-ISF asks

> Is there a satisfiable $\psi(\vec{x})$ such that $\psi \models \phi$ and ψ is semantically invariant under G? Formally, is it the case that $(\exists a \in \mathsf{Asg}(\vec{x}))(\forall g \in G)\ \phi^g(a) = 1$ is true? Or alternatively, is $\bigwedge_{g \in G} \phi^g$ satisfiable?

In terms of models of ϕ we are asking if there is a nonempty subset of the models which is closed under G-action. We show in the following theorem that the problem is Σ_2^p-complete.

Theorem 7.3.1 G-ISF is Σ_2^p-complete.

Proof This problem has a slightly different format from the problems met so far in that the quantification over assignments and group elements is reversed. It is straightforward to see that the problem is in Σ_2^p following the same encoding principles as in Theorem 5.3.1. As in Theorem 7.2.1 for G-EMBED we show a transformation from B_2 with instance $\phi(\vec{x}, \vec{y})$, the standard complete problem for Σ_2^p of Theorem 2.4.2 to G-ISF. Again we use $\langle a, b \rangle$ with $a \in \mathsf{Asg}(\vec{x})$ and $b \in \mathsf{Asg}(\vec{y})$ to range over elements of $\mathsf{Asg}(\vec{x}, \vec{y})$. This time we apply the Δ-Asg correspondence to the inner quantification, the universal one, and we exploit the fact that for $\delta \in \Delta(\vec{y})$ then $\phi(\langle a, b^\delta \rangle) = \phi(\langle a, b \rangle^\delta) = \phi^\delta(\langle a, b \rangle)$ since $a = a^\delta$ (δ only moves points in $\vec{y}$).

$$
\begin{aligned}
(\exists \vec{x})(\forall \vec{y}) \models \phi \quad &\Leftrightarrow \quad (\exists a \in \mathsf{Asg}(\vec{x}))(\forall b \in \mathsf{Asg}(\vec{y}))\ \phi(\langle a, b \rangle) = 1 \\
&\Leftrightarrow \quad (\exists a \in \mathsf{Asg}(\vec{x}))(\exists b \in \mathsf{Asg}(\vec{y}))(\forall \delta \in \Delta(\vec{y}))\ \phi(\langle a, b^\delta \rangle) = 1 \\
&\Leftrightarrow \quad (\exists \langle a, b \rangle \in \mathsf{Asg}(\vec{x}, \vec{y}))(\forall \delta \in \Delta(\vec{y}))\ \phi^\delta(\langle a, b \rangle) = 1 \\
&\Leftrightarrow \quad (\exists c \in \mathsf{Asg}(\vec{x}, \vec{y}))(\forall \delta \in \Delta(\vec{y}))\ \phi^\delta(c) = 1 \\
&\in \quad \text{G-ISF}
\end{aligned}
$$

We can rewrite G-ISF instances for wff $\phi(\vec{x})$ in the terms of the satisfiability of $\bigwedge_{g\in G}\phi^g$ as follows where $G = \{g_1, \ldots, g_n\}$.

$$\begin{aligned}(\forall g \in G)(\phi^g(a) = 1) &\Leftrightarrow \phi^{g_1}(a) = 1 \ \& \ \ldots \ \& \ \phi^{g_n}(a) = 1\\ &\Leftrightarrow (\phi^{g_1} \wedge \ldots \wedge \phi^{g_n})(a) = 1\\ &\Leftrightarrow (\bigwedge_{g\in G}\phi^g)(a) = 1\end{aligned}$$

Therefore $\bigwedge_{g\in G}\phi^g$ is satisfiable if and only if $(\exists a \in \mathsf{Asg}(\vec{x})(\forall g \in G)\ \phi^g(a) = 1$. The D-transform may be applied to show that subgroups of symmetric group on the variables of ϕ give equally hard instances of this problem. Recall that $D(\phi) \stackrel{\text{def}}{=} \phi \wedge \zeta$ where ζ is a system of definitions in the form of a 2-CNF. We need a new fact about D namely $D(\phi \wedge \psi) \stackrel{\text{def}}{=} (\phi \wedge \psi) \wedge \zeta \equiv (\phi \wedge \zeta) \wedge (\psi \wedge \zeta) = D(\phi) \wedge D(\psi)$. Using Lemma 3.4.2 and the fact that $D(\phi)$ is satisfiable if and only if ϕ is (since their models are in 1-1 correspondence) we have:

$$\begin{aligned}\textsc{Sat}(\bigwedge_{g\in G}\phi^g) &\Leftrightarrow \textsc{Sat}(\phi^{g_1} \wedge \ldots \wedge \phi^{g_n})\\ &\Leftrightarrow \textsc{Sat}(D(\phi^{g_1} \wedge \ldots \wedge \phi^{g_n}))\\ &\Leftrightarrow \textsc{Sat}(D(\phi^{g_1}) \wedge \ldots \wedge D(\phi^{g_n}))\\ &\Leftrightarrow \textsc{Sat}(D(\phi)^{D(g_1)}) \wedge \ldots \wedge D(\phi^{D(g_n)}))\\ &\Leftrightarrow \textsc{Sat}(\bigwedge_{d\in D(G)} D(\phi)^d)\end{aligned}$$

Hence any instance of G-ISF may be transformed to an instance quantifying over subgroup of a symmetric group. □

In the following theorem we summarise some of the different cases for G-ISF for various classes of formula.

Theorem 7.3.2 G-ISF is:

i). Σ_2^p-complete for $\phi \in$ 3-DNF (or any generalisation);

ii). NPC for $\phi \in k$-CNF for fixed $k \geq 3$;

iii). polynomial for 2-CNF;

iv). polynomial for k-Horn.

Proof

i). Follows directly from B_2 reduction of Theorem 7.3.1 where 3-DNF is sufficient for Σ_2^p-completeness by Theorem 2.4.2.

ii). Given $G \leq W(\vec{x})$ then $\bigwedge_{g \in G} \phi^g$ can be computed in polynomial time essentially because the number of distinct clauses in the conjunction cannot exceed $O(n^k)$ where $n = |\vec{x}|$. Let $C_0, C_1, \ldots, C_{n-2}$ be transversals for a stabiliser chain for G as described in §3.6. Then

$$\bigwedge_{g \in G} \phi^g = \bigwedge_{c_0 \in C_0} (\bigwedge_{c_1 \in C_1} \cdots (\bigwedge_{c_{n-2} \in C_{n-2}} \phi^{c_{n-2}}) \ldots)^{c_1})^{c_0}$$

In other words, working from the interior of this product we compute the conjunction of $O(n)$ images of ϕ and then delete any repeated clauses. Each of the $O(n)$ steps is polynomial since the number of clauses never exceeds $O(n^k)$. Finally we have to check whether the resulting formula is satisfiable. So the G-ISF problem polynomially transforms to satisfiability in this case. It is NP-hard by just sending a typical SAT instance (3-CNF is sufficient) to G-ISF with the trivial group 1. Therefore the restricted problem is NPC.

iii). Follows the argument for ii) except that the final satisfiability check is also polynomial

iv). G-ISF for ϕ in k-Horn is polynomial by the same argument as for iii) unless G is not a subgroup of $S(\vec{x})$ in which case we apply the D-transform to the problem as in the above discussion. This is necessary to ensure that the expanded formula $\bigwedge_{g \in G} \phi^g$ is still in Horn form, which might not be the case if G contains any elements which affect complementarity. Note that the formula ζ added in the D-transform is in 2-Horn so that if ϕ is in k-Horn then so in $D(\phi)$. And note that since $D(G)$ is a subgroup of a symmetric group then $\bigwedge_{d \in D(G)} D(\phi)^d$ is still in k-Horn. Hence even after computing the conjunction in polynomial time we still have a polynomial satisfiability check for the resulting formula.□

The observation that G-ISF can be viewed as the question of whether a subset of the models of ϕ are closed under G-action leads to the complementary question of whether there are no such closed subsets. This has a relationship to the problem of constructing partial symmetry-breaking predicates discussed in §4.6 as follows. If there are no closed subsets of the set of models, this is saying that every G-orbit of the assignments contains at least one countermodel of ϕ. Equivalently, every G-orbit of the the assignments contains at least one model of $\neg\phi$. Equivalently, $\neg\phi$ is a partial symmetry-breaking predicate for the group G: it is true on at least one assignment from every G-orbit of the assignments. Deriving the complementary problem from formal negation of the G-ISF statement we obtain the question of whether the following is valid:

$$\bigvee_{g \in G} \neg\phi.$$

We know from (i) of Theorem 7.3.2 that G-ISF is Σ_2^p-complete for 3-DNF formulas. Applying negation to ϕ yields a 3-CNF formula and hence G-ISF COMPLEMENT is Π_2^p-complete for 3-CNF formulas. Hence the decision problem for 3-CNF (or generalisation) $\phi(\vec{x})$ and group $G \leq S(\vec{x})$ phrased as whether ϕ is a partial symmetry-breaking predicate for group G is Π_2^p-complete.

7.4 The $\mathcal{F}$-invariant subformula problem

This section examines the complexity of the $\mathcal{F}$-invariant subformula problem ($\mathcal{F}$-ISF): given wff $\phi(\vec{x})$ and a family $\mathcal{F}$ of subgroups $G \leq W(X)$ specified by a polynomial time membership test, the problem $\mathcal{F}$-ISF asks:

> Is $\langle G, \phi \rangle$ a member of G-ISF for some $G \in \mathcal{F}$?

Recall that G will always be given in the form of a list of generators and that polynomial time verification of membership in $\mathcal{F}$ is based on this representation. We show that the problem is Σ_2^p-complete.

Theorem 7.4.1 $\mathcal{F}$-ISF is Σ_2^p-complete

Proof The problem is in Σ_2^p since we have just added another existential quantification to the front of the question, and we have ensured that $G \in \mathcal{F}$ is checkable in polynomial time. See the membership criteria for Σ_2^p of Theorem 2.4.1 for clarification. It is therefore Σ_2^p-complete since it is a generalisation of G-ISF: for any instance $\langle G, \phi \rangle$ of G-ISF we could define $\mathcal{F} = \{G\}$ giving a transformation from G-ISF to $\mathcal{F}$-ISF. Membership in this 'family' is checkable in polynomial time since equality of groups given by possibly different lists of generators is still polynomial time, as discussed in §3.6.□

An example family that we consider in Chapter 8 where this problem is discussed in relation to graph-theoretic problems is the family of all transitive cyclic groups on elements $\vec{x}$, that is all groups generated by a single element which consists of a single cycle containing all of $\vec{x}$ in some order.

7.5 The C-disjunction problem

This section examines the C-disjunction problem for boolean formulas (C-DISJ): given wff $\phi(\vec{x})$ and coset $C = Gk$ with $G \leq W(\vec{x})$ and $k \in W(\vec{x})$ the problem C-DISJ asks:

> Is the union of ϕ and some image of ϕ under C valid? Formally, is it the case that $(\exists g \in C) \models \phi^g \vee \phi$.

The main point of interest concerning this problem is that it is an example of a Σ_2^p-complete problem similar to G-EMBED which contains only one formula. It can be viewed as a containment problem by putting $\phi^g \vee \phi \equiv (\neg\phi)^g \rightarrow \phi$. In other words we are asking equivalently if the complement of ϕ can be embedded in ϕ. The problem arose out of an attempt to apply the same techniques which showed that the G-EQUIV and C-INVAR problems of the previous chapter were polynomially equivalent. The G-EQUIV problem has two formulas and the C-INVAR problem has just one.

Theorem 7.5.1 C-DISJ in Σ_2^p-complete.

Proof We show that this problem is polynomially equivalent to G-EMBED which is Σ_2^p-complete by Theorem 7.2.1. To show C-DISJ $\leq_p$ G-EMBED we can put

$$\begin{aligned}(\exists g \in Gk) \models \phi^g \vee \phi &\Leftrightarrow (\exists j \in G) \models \phi^{jk} \vee \phi \\ &\Leftrightarrow (\exists j \in G) \models \phi^j \vee \phi^{k^{-1}} \\ &\Leftrightarrow (\exists j \in G) \models (\neg\phi)^j \rightarrow \phi^{k^{-1}} \\ &\in \text{G-EMBED}\end{aligned}$$

To show G-EMBED $\leq_p$ C-DISJ we use the disjunctive version Lemma 3.3.1 of the gluing technique of §3.3 to reduce any instance of G-EMBED to an instance of C-DISJ in polynomial time. We construct an instance of C-DISJ using the coset $(Gh^{-1}Gh)h^{-1} = Gh^{-1}G$ of $Gh^{-1}Gh$ where h is the renaming permutation of the gluing technique.

$$\begin{aligned}(\exists g \in G) \models \phi^g \rightarrow \psi &\Leftrightarrow (\exists g \in G) \models (\neg\phi)^g \vee \psi \\ &\Leftrightarrow (\exists \gamma \in Gh^{-1}G) \models (\neg\phi \vee \psi^h)^\gamma \vee (\neg\phi \vee \psi^h) \\ &\in \text{C-DISJ}\end{aligned}$$

To see this note that for $g, j \in G$

$$(\neg\phi \vee \psi^h)^{gh^{-1}j} \vee (\neg\phi \vee \psi^h) \equiv (\neg\phi^g \vee \psi)^h \vee (\neg\phi \vee \psi^j)$$

and by Lemma 3.3.1 RHS is valid if and only either $\models \neg\phi^g \vee \psi$ or $\models \neg\phi \vee \psi^j$. Now this is equivalent to our original instance of G-EMBED since if this holds for some g' then the first of the above expressions is true of g' and if one or other of the above expressions is true then either g or j^{-1} is true for the G-EMBED instance.

Note that by the reduction from G-EMBED and the observation in the proof of this problem's Σ_2^p-completeness even for subgroups of the symmetric group on variables, then C-DISJ must also be Σ_2^p-complete under this restriction.□

The nice aspect of this result is that it addresses the question of the hardness of Σ_2^p-complete embedding problems being dependent on two formulas in the problem instance. Say one is given G, ϕ in an instance of G-EMBED: then it may always be possible to find a ψ that makes the problem difficult. This extra degree of freedom is removed in C-DISJ so that formula/group pairs can be intrinsically hard.

7.6 Summary

We have presented four problems relating groups and formulas and shown that in the general case they are Σ_2^p-complete. We have also shown that certain special cases admit better upper bounds. In the next chapter we discuss these results in conjunction with similar results concerning the equivalence problems of the previous chapter. Some unresolved cases of this chapter are summarised in §8.7.

Chapter 8

Discussion

8.1 Introduction

This chapter examines some of the issues raised by the previous two chapters. In particular we examine some issues that are more speculative in nature and which constitute a selection of areas not resolved by this thesis, but which appear to be of interest. In this respect the chapter is an extended preface to the concluding remarks of the final chapter. Three conjectures are presented concerning problems left open by the previous two chapters.

8.1.1 Plan of chapter

In §8.2 we return to the question of why the unbounded CNF formula seem to present harder equivalence and containment problems than the formulas with clause size bounded by some fixed number k. There is a classical method of reducing formulas in CNF to formulas in 3-CNF preserving some important properties. We examine why this method does not seem to work directly in the context of preserving equivalence or containment under group action. This means that a reduction to 3-CNF cannot apparently be employed as a preprocessing technique to make the problems easier. We conjecture that the restriction to CNF formulas is as hard as any other case for the problem G-EQUIV, having shown this to be true for G-EMBED (Lemma 7.2.2).

An important question is whether the problem G-EQUIV and related equivalence prob-

lems are Σ_2^p-complete. In §8.3 we look at two pieces of evidence that they are not Σ_2^p-complete, and we conjecture that a third piece of evidence is available through an analogy with the interactive proof approach to showing that GRAPH ISOMORPHISM is not NPC. Constructing this analogy seems to present some interesting problems.

Issues concerning the containment problems of Chapter 7 are discussed next in §8.4 §8.5. We argue that problems G-EMBED and $\mathcal{F}$-ISF can be viewed as natural complete problems for the class Σ_2^p by viewing them as harder versions of graph-theoretic problems. We conjecture the existence of a relationship between 'hard' groups for the problem $\mathcal{F}$-ISF and for the corresponding natural graph-theoretic problem. We argue that the containment problems are probably not useful for establishing Σ_2^p-completeness of other suspected candidates for this class as a result of their generality.

Two summaries are given of Chapters 6 and 7. The first in §8.6 gives a table of resolved cases which presents nicely the range of complexities for various formulations of problems concerning groups acting on formulas. This is in addition to the summaries for the special cases for representative problems G-EQUIV and G-EMBED of Figures 6.1 and 7.1. A second summary §8.7 completes the picture by reviewing a number of the special cases that we have not resolved.

8.2 CNF seems harder than k-CNF

In this section we discuss why it appears that for the problems G-EQUIV and G-EMBED of Chapters 6 and 7 respectively we get harder instances with CNF formulas of *unbounded* clause size. Recall Lemma 7.2.2 which established that if a restriction was applied to G-EMBED such that both formulas must be in CNF then the problem was still Σ_2^p-complete, whereas Theorem 7.2.5 established that what is believed to be a strictly lower bound applies if the formulas are in k-CNF for some fixed k. One interesting point here is that the complete problems B_k for each level of the polynomial hierarchy have been shown sufficiently hard for 3-τNFs where $\tau = C$ if k is odd or $\tau = D$ if k is even. It seems as though the more structural questions arising from group action are not well preserved by operations which manipulate formulas into

3-τNF, an example of which we examine below.

We have an upper bound of Σ_2^p for G-EQUIV in the case of two arbitrary wffs by Theorem 5.3.1. We have an upper bound of $\mathrm{P}^{\mathrm{NP}[\log n]}$ if one of the formulas is in k-CNF by Theorem 6.2.3 since we can construct closed forms for both formulas, reducing the problem to OPS. One mystery is how hard G-EQUIV is for free CNF. In this section we illustrate why the usual polynomial time method for turning a free CNF into a 3-CNF (preserving certain properties) does not seem to provide a better worst case analysis for G-EQUIV on CNFs.

8.2.1 3-CNF from CNF

The standard technique for constructing a 3-CNF formula ψ from a CNF ϕ proceeds as follows. Take a clause $c = l_1 \vee \ldots \vee l_i$ in ϕ and a new variable x and replace c initially with

$$(l_1 \vee l_2 \vee x) \wedge (x \leftrightarrow (l_3 \vee \ldots \vee l_i)). \tag{8.1}$$

The term on the right can be replaced with the equivalent

$$(\neg x \vee l_3 \vee \ldots \vee l_i) \wedge (\neg l_3 \vee x) \wedge \ldots \wedge (\neg l_i \vee x) \tag{8.2}$$

and the process is repeatedly applied to the first of these new clauses until we have a system of clauses of size 2 or 3 representing the original clause c. This leads to $\Theta(i^2)$ new clauses and $\Theta(i)$ auxiliary variables per clause. Applying this procedure to every clause in ϕ results in a formula ψ which can be constructed in polynomial time. Also quite a lot of the structure has been preserved by the construction. The models of ϕ over $\mathsf{Vars}(\phi)$ are in 1-1 correspondence with the models of ψ over $\mathsf{Vars}(\psi)$ as a result of the sequence of definitions of the auxiliary variables in the construction process. That is, every model of ϕ extends to a unique model of ψ, and every model of ψ is also a model of ϕ.

Unfortunately the structure we are interested in is likely to have been lost. Call the transformation on formulas T. The basic problem is that there is does not seem to be any way of defining T on group elements such that in general $T(\phi^g) \equiv T(\phi)^{T(g)}$. We

were able to do this (Lemma 3.4.2) with the D-transformation, which employs a similar method of defining auxiliary variables in terms of old ones, because the definitions were of the form $x \leftrightarrow \neg x'$, in terms of just one other literal. We can say that in the general case it is 'very improbable' that T can be defined on group elements such that we can apply transformation to give

$$\models \phi^g \rightarrow \psi \quad \Leftrightarrow \quad \models T(\phi)^{T(g)} \rightarrow T(\psi)$$

where $T(\phi)$ and $T(\psi)$ are in 3-CNF since this would give a reduction of Σ_2^p-complete CNF case to the $\mathrm{P}^{\mathrm{NP}[\log n]}$ k-CNF case.

We look at what *can* be said about the constructed 3-CNF formulas in relation to group action, and this will illustrate the flaw in attempting the above reduction. We split the construction of 3-CNF for ϕ into two parts ϕ' and ϕ'' where ϕ' consists of all the initial 3-clauses with just one auxiliary variable (the left part in (8.1) for each clause) and ϕ'' consists of system of definitions of the auxiliary variables (all the bits in (8.2) for each clause). Note that when applying to different formulas, new auxiliary variables must be used. Then it can be shown in similar fashion to the proof of Lemma 3.4.1 that for any ϕ, ψ in CNF

$$\begin{aligned} \models \phi \rightarrow \psi \quad &\Leftrightarrow \quad \models (\phi' \wedge \phi'') \rightarrow (\psi \wedge \phi'') \\ &\Leftrightarrow \quad \models (\phi' \wedge \phi'' \wedge \psi'') \rightarrow (\psi' \wedge \psi'' \wedge \phi'') \end{aligned}$$

where all formulas are now in 3-CNF. What we have ensured is that when adding definitions to one side, we have added the same definitions to the other side of the implication. However, if we consider how the correspondence could be applied to instances of G-EMBED for free CNF formulas then we get

$$(\exists g \in G) \models \phi^g \rightarrow \psi \quad \Leftrightarrow \quad (\exists g \in G) \models (\phi'^g \wedge \phi''^g \wedge \psi'') \rightarrow (\psi' \wedge \psi'' \wedge \phi''^g)$$

with RHS not being a recognisable instance of G-EMBED, but of another apparently quite different problem. Although the six formulas of the new problem are now in 3-CNF, it does not seem to be helpful to apply the $M_{3,C,\vec{x}}$ construction to these formulas. In other words, no preprocessing using $M_{3,C,\vec{x}}$ seems to yield a problem in NP, as was

the case in Theorem 6.2.3. And this seems to be because union of two closed formulas, corresponding to conjunction of CNFs, does not give a closed formula (intersection does).

This seems to illustrate why the standard 3-CNF transform does not map G-EMBED problems to G-EMBED problems. And we have seen that this approach is very unlikely to work anyway from the consequence $\Sigma_2^p \subseteq \mathrm{P}^{\mathrm{NP}[\log n]}$ that would be implied.

Note The implications in the above statements can be replaced with double implications by combining both directions.

Conjecture 8.2.1 We conjecture that G-EQUIV restricted to pairs of CNFs with unbounded clause size gives the hardest instances for that problem.

8.3 Non Σ_2^p-completeness of the equivalence problems

Is G-EQUIV Σ_2^p-complete? First we look at two important pieces of evidence that suggests the problem is not Σ_2^p-complete and then we look more speculatively at some of the evidence that GRAPH ISOMORPHISM is not NPC, and how that evidence might be transferred up one level of the hierarchy to the G-EQUIV problem.

8.3.1 Does SAT transform to G-EQUIV?

It does not seem to be possible to show that SAT polynomially transforms to G-EQUIV, even though, in particular for arbitrary wffs, G-EQUIV appears to be a hard problem lying outside of NP and co-NP, as was established in Theorem 6.2.1. On the other hand we showed that a *randomised* polynomial transformation of SAT to G-EQUIV can be constructed. This difficultly is not exceptional for problems that appear to be actually outside of NP $\cup$ co-NP. We met the UNIQUESAT problem in §6.2.1. This problem has also not been shown to be hard for NP with respect to polynomial transformations. Note that if SAT could be polynomially transformed to UNIQUESAT then it could also be transformed to G-EQUIV by (iii) of Theorem 6.2.1. Now any Σ_2^p-complete problem must be both NP and co-NP hard with respect to polynomial transformations. We

can transform both SAT and UNSAT to the problem B_2 of Theorem 2.4.2 by ignoring respectively the $\forall$ and $\exists$ parts.

Therefore in the eventuality that SAT can be proved irreducible to G-EQUIV, then the latter problem is definitely not Σ_2^p-complete. However this implication only goes one way. There are 'simple' problems in Σ_2^p which are hard for both NP and co-NP with respect to polynomial transformation but which are unlikely to be Σ_2^p-complete. For example SAT/UNSAT.

A further point transfers this argument to the $\mathrm{P}^{\mathrm{NP}[\log n]}$ result of Theorem 6.2.3 for G-EQUIV restricted to k-CNF formulas for fixed k (or just one of the formulas in fact). Clearly SAT is contained in $\mathrm{P}^{\mathrm{NP}[1]}$. Hence the difficulty of reducing SAT to G-EQUIV in general would suggest the difficulty of showing that the restricted k-CNF case is complete for $\mathrm{P}^{\mathrm{NP}[\log n]}$.

8.3.2 Counting argument

We can use standard results about the difficulty of counting solutions to problems to provide evidence that G-EQUIV is easier than the Σ_2^p-complete problem G-EMBED. We discussed how SAT reduces to B_2 above. It is also easy to show a transformation of SAT into the problem G-EMBED which preserves the number of solutions. If $\vec{x} = \mathsf{Vars}(\phi)$ then

$$\textsc{Sat}(\phi) \quad \Leftrightarrow \quad (\exists \delta \in \Delta(\vec{x})) \models (\wedge \vec{x})^{\delta} \rightarrow \phi$$

We used a corresponding argument in Theorem 6.2.1 to show transformation from UNIQUESAT to G-EQUIV. The above transformation is 'parsimonious' in the sense that the number of models of ϕ over $\vec{x}$ is the same as the number of δ which embed the minterm $\wedge \vec{x}$ in ϕ. This means that the counting problem for G-EMBED is at least as hard as the counting problem for SAT (called #SAT). It is known that a polynomial time DTM with an oracle for #SAT can recognise any language in the polynomial hierarchy (Toda's theorem [Tod89]). Therefore an oracle for counting solutions to G-EMBED would also suffice to recognise any language in the polynomial hierarchy. However we saw in §6.5.1 that the order of the semantic symmetry group of ϕ can be

computed in the third level of the hierarchy, namely Δ_3^p. It is an easy lemma that the number of maps g in a group G such that $\models \phi^g \leftrightarrow \psi$ is either zero or equal to the number of semantic symmetries of ϕ in G (there is at least one map g if and only if there is a different map hg for every semantic symmetry h of ϕ). Hence the counting problem for G-EQUIV is also in Δ_3^p. Therefore we cannot recognise all languages in the polynomial hierarchy using the counting problem for G-EQUIV as an oracle unless the polynomial hierarchy is contained in Δ_3^p (note that $\mathrm{P}^{\Delta_3^p} = \Delta_3^p$), something that is of the same order of unlikelyhood as showing P = NP.

This provides evidence that the counting problem for G-EMBED is strictly harder than than the counting problem for G-EQUIV and this suggests that G-EMBED is strictly harder than G-EQUIV which would therefore not be Σ_2^p-complete.

Note This argument follows the same lines as similar arguments for the non-NP-completeness of GRAPH ISOMORPHISM, first observed by Mathon [Mat79].

8.3.3 The graph analogy

The GRAPH ISOMORPHISM problem has resisted all attempts at classification as either a problem in P or an NPC problem. Garey and Johnstone have observed [GJ79] that there is something about the graph isomorphism problem which seems to prevent transformation of any known NPC problems into it. Their observation is that there is a lack of redundancy in the problem. Any modification of one of the graphs results in a destruction of the property being isomorphic, whereas this is not the case for an NPC problem like SUBGRAPH ISOMORPHISM. The property of embedding for graphs is not affected by adding edges to the larger graph, or subtracting edges from the smaller graph. Problem reductions seem to require a degree of redundancy in the target problem, but the lack of this property in GRAPH ISOMORPHISM has yet to yield any formal reason why it is impossible to reduce an NPC problem to it.

Evidence that it is *not* NPC has however come from a different direction. Goldwasser et al. showed that the complement of the graph isomorphism problem has short interactive

proofs, in fact that the problem was in their class[1] IP[2] [GMR85]. The protocol which recognises the language GRAPH ISOMORPHISM COMPLEMENT is as follows. The verifier 'Bob' chooses randomly and secretly one of the two input graphs and applies a random permutation to the vertices. He then presents the graph to the prover 'Alice' who then has to decide which of the original graphs it is. If they are not isomorphic then Alice will always be able to give the right answer. If they are isomorphic then Alice can be right with probability no more than 1/2. Using amplification (repeating the protocol) reduces the error bound to under a 1/3 as required.

Babai introduced an interactive proof class[2] AM [Bab85] which was shown later in [GS88] to define the same languages as IP[k] for any fixed k. The advantage of this equivalent class, despite less intuitive and seemingly more restricted protocols, is that complexity results concerning it are easier to establish. Boppana et al. showed that if any co-NPC problem were in AM then the polynomial hierarchy would collapse to the second level, i.e. for all $k \geq 2$, $\Sigma_k^p = \Pi_k^p = \text{AM}$ [BHZ87]. The general view seems to be that this is of the same order of unlikelyhood as proving P = NP. This is therefore strong evidence that GRAPH ISOMORPHISM COMPLEMENT is not co-NPC (and so GRAPH ISOMORPHISM is not NPC) because, as we noted above, Goldwasser et al. showed that the problem is in IP[2] = AM.

We propose that similar methods could be applied to the G-EQUIV problem to show that it is not Σ_2^p-complete. We sketch out an argument here. What appears to be interesting about this approach is that what should correspond to the correct generalisation of 'short interactive proof' to capture problems one level higher in the polynomial hierarchy does not seem to be strong enough to provide protocols for recognising G-EQUIV COMPLEMENT.

Note that we showed that G-EQUIV was co-NP-hard with respect to polynomial transformation in Theorem 6.2.1. Therefore it is unlikely that G-EQUIV is in AM by the

[1] The bracketed number refers to the number of rounds of message-passing allowed in the protocol. A good introductory paper on complexity classes arising from interactive proof systems is [Joh92].

[2] The class MA $\subseteq$ AM encountered in §6.6 is the dual class obtained by reversing the order of the protocol in AM.

above argument of Boppana et al.

It seems to be necessary to give Babai's verifier 'Arthur' more power. The obvious step seems to be to provide him with an oracle for SAT. We propose the class $\mathrm{A^{NP}M}$ where protocols are exactly the same as for AM, except that Arthur may use the SAT oracle to reach his decision in polynomial time after the prover 'Merlin's reply. It follows straightforwardly from the argument that NP is contained in AM (a slight randomised extension of NP was Babai's aim in defining the class AM) that Σ_2^p is contained in $\mathrm{A^{NP}M}$: Merlin provides a witness for the existential part of B_2 of Theorem 2.4.2, ignoring Arthur's message, and Arthur verifies the universal part with the aid of the oracle. Therefore this new class is a randomised extension of Σ_2^p.

Recall Conjecture 8.2.1 which proposed that hard instances for G-EQUIV consist of pairs ϕ, ψ of CNF formulas. Consider the problem G-EQUIV COMPLEMENT to establish that there are *no* maps in some given group G which send CNF ϕ to CNF ψ. We conjecture the following based on the above discussed result that AM is unlikely to contain co-NP and so GRAPH ISOMORPISM is not NPC.

Conjecture 8.3.1 The following hold:

i). $\mathrm{A^{NP}M}$ does not contain Π_2^p.

ii). G-EQUIV COMPLEMENT is in $\mathrm{A^{NP}M}$.

By (i) and (ii) it would follow that G-EQUIV is not Σ_2^p-complete. Although this seems to be a reasonable conjecture by analogy with the graph case, it seems to be difficult to establish part (ii). The underlying idea in the graph case is that it if the two graphs are isomorphic then there is a map g on vertices such that $E_1^g = E_2$. Therefore it is impossible for the prover to distinguish between the graphs arising from gh applied to E_1 or h applied to E_2. However in the case of two CNF formulas such that $\models \phi^g \leftrightarrow \psi$ then gh applied to ϕ is *logically* indistinguishable from h applied to ψ, but inspection may easily reveal to the prover which formula has been used and which map has been applied, since the two formulas may not be syntactically equivalent under the group.

This leads to thought-experiments along the lines of whether the randomly chosen formula can be scrambled in some way, preserving equivalence and in polynomial time (with SAT oracle available), by the verifier after he has applied the random map to it, such that it is not possible for the prover to link it with certainty to one of the originals unless the two formulas are inequivalent under the group. The difficulty here is the unlimited resources the prover is allowed in descrambling the formula. We leave this problem open.

Note that in the restricted cases for G-EQUIV shown to lie in $\mathrm{P}^{\mathrm{NP}[\log n]}$ or below then their complements are easily seen to be in $\mathrm{A}^{\mathrm{NP}}\mathrm{M}$ since the latter class is closed under complementation and is contained in Σ_2^p and therefore in $\mathrm{A}^{\mathrm{NP}}\mathrm{M}$.

8.4 Naturality of the containment problems

If the naturality of a problem is determined to be related to the practical significance of its complexity in some established algorithmic domain then we do not have much to offer in the way of arguments for naturality of G-EMBED and G-ISF as hard representatives of the class Σ_2^p. (We did however show in §7.3 that the partial-symmetry breaking predicate construction of [CGLA96] discussed in §4.6 gave rise to a Π_2^p-complete decision problem.)

However we show in this section that these problems are closely related to natural problems that arise in graph theory and we argue that their naturality arises from this correspondence.

Recall the interesting relation between G-EMBED and G-ISF in that they are in some sense dual, arising from reductions which make use of the Δ-Asg correspondence in two different ways (Theorems 7.2.1 and 7.3.1). Recall also the problem $\mathcal{F}$-ISF which is G-ISF with an extra layer of quantification, namely over a family $\mathcal{F}$ of groups, which was also shown to be Σ_2^p-complete (Theorem 7.4.1).

By the graph to formula constructions of §3.5.3 then the NPC graph problems that we discuss in this section are easily reducible to instances of G-EMBED and $\mathcal{F}$-ISF,

giving a little weight to the naturality of these problems as harder versions of the same general notions. In this section we compare these problems with well-known graph-theoretic problems that can be phrased either in terms of embedding problems or (not seen elsewhere) in terms of whether a graph contains a subgraph invariant under some group from a given family $\mathcal{E}$. We conclude the section with a conjecture that a correspondence can be established between the graph and formula cases which could provide some interesting results.

Consider the NPC problem HAMILTONIAN CIRCUIT for a graph $H = (V, E)$:

> Does H contain a circuit of all the vertices, visiting no vertex more than once and returning to the same point?

This can be reformulated (standardly) as a SUBGRAPH ISOMORPHISM problem:

> Does H contain a subgraph isomorphic to the cyclic graph over $n = |V|$ vertices?

Furthermore, if we let $\mathcal{E}$ be the family of transitive cyclic groups over V (those generated by a single cycle containing all of V), then we can ask equivalently

> Does H contain a nontrivial subgraph invariant under some $G \in \mathcal{E}$?

Note that this problem is in NP: having guessed the generator of G, we need to check

$$\bigcap_{g \in G} E^g \neq \{\}$$

which can be done in polynomial time. The details are similar to the technique of ii) of the proof of Theorem 7.3.2 (so in fact the check is polynomial for any $G \leq S(V)$ not just cyclic groups.)

The NPC problem CLIQUE

> Does H contain a complete graph of k vertices?

can similarly be reduced to both types of question. In the first (standard) case we just ask whether H contains a subgraph isomorphic to the complete graph with k vertices; in the second case we consider the familiy $\mathcal{E}$ of symmetric subgroups of $S(V)$ of degree k. The only nontrivial graph which is invariant under a symmetric group is the complete graph on the same points.

A third NPC example which can be reduced to the invariant subgraph problem is FIXED POINT FREE AUTOMORPHISM (e.g. see [Lub81])

Does $\mathsf{Aut}(H)$ contain a permutation which moves every vertex?

Here we consider the family $\mathcal{E}$ of all cyclic groups on V which move all points, i.e. those generated by a single permutation whose cycle representation contains all vertices[3].

So we have a number of natural graph-theoretic questions expressed in terms of either embedding problems or problems involving whether they have subgraphs invariant under some group belonging to a given family.

We conjecture a correspondence can be defined between families $\mathcal{E}, \mathcal{F}$ of subgroups respectively of $S(V)$ and $W(\vec{x})$ such that the NPC invariant subgraph problems with family $\mathcal{E}$ translate from and to Σ_2^p-complete problems for $\mathcal{F}$-ISF with family $\mathcal{F}$. In the case of HAMILTONIAN CIRCUIT or FIXED POINT FREE AUTOMORPHISM then the associated families of cyclic permutation groups do not give rise to hard instances of $\mathcal{F}$-ISF. Restricted to families of permutation groups of polynomial-bounded order, then $\mathcal{F}$-ISF is NPC since the term

$$\bigwedge_{g \in G} \phi^g$$

expands to a formula of polynomial size. Thus we can guess a group $G \in \mathcal{F}$ and guess an assignment $a \in \mathsf{Asg}(\vec{x})$ and check that a satisfies the above term in polynomial time. Hence the problem is in NP. It is NPC since we can reduce a typical SAT instance to the problem with the family $\{1\}$. Hard families for graph problems are therefore not

[3] It is interesting how the two apparently dissimilar problems HAMILTONIAN CIRCUIT and FIXED POINT FREE AUTOMORPHISM are closely analogous in this interpretation: the only difference is that in HAMILTONIAN CIRCUIT we are restricted to transitive cyclic groups.

necessarily hard for $\mathcal{F}$-ISF. The problem is to see what the family of groups $\mathcal{E}$ would translate to in the formula case. From the B_2 reduction to G-ISF of Theorem 7.3.1 we see that one example hard family of groups would be any group containing $\Delta(\vec{y})$ for $\vec{y} \subseteq \vec{x}$. Now $\Delta(\vec{y})$ can be viewed as the base subgroup of $S_2 \wr C(\vec{y})$ for the cyclic group on $\vec{y}$. The correspondence which maps a group G for invariant subgraph problems to $S_2 \wr G$ for G-ISF would perhaps be a natural candidate to consider, and completeness would hold in this direction for the example invariant subgraph problems given above. But that leaves the question of how arbitrary hard families for $\mathcal{F}$-ISF would translate back to families $\mathcal{E}$ in the graph case.

A bidirectional correspondence between families $\mathcal{E}, \mathcal{F}$ which are hard for respectively the invariant subgraph and invariant subfunction problems would be extremely interesting because it might for example yield a simple proof of the NP-completeness of HAMILTONIAN CIRCUIT. (These proofs tend to run to many pages of detailed graph constructions which are interesting in their own right, but it would be nice also to have a succinct proof.) In other words, since the underlying problem G-ISF is already Σ_2^p-complete, whereas for any group it is polynomial to check whether a graph contains a subgraph invariant under the group, then it might be much easier to show that the corresponding family $\mathcal{F}$ gives a Σ_2^p-complete problem for $\mathcal{F}$-ISF where $\mathcal{E}$ is the family of cyclic groups associated with HAMILTONIAN CIRCUIT, and then to deduce from the correspondence that $\mathcal{E}$ must give an NPC graph problem. We summarise this idea as follows.

Conjecture 8.4.1 There is a relationship between sets of vertices V and sets of propositional variables $\vec{x}$ and between families $\mathcal{E}, \mathcal{F}$ of subgroups respectively of $S(V)$ and $W(\vec{x})$ such that

$$(\exists G \in \mathcal{E}) \bigcap_{g \in G} E^g \neq \{\}$$

is an NPC problem if and only if

$$(\exists G \in \mathcal{F}) \, \text{SAT}(\bigwedge_{g \in G} \phi^g)$$

is a Σ_2^p-complete problem.

8.5 Generality of the containment problems

In the appendices to [GJ79], Garey and Johnson give a large number of established NPC problems with details of how their completeness proofs have been constructed by reductions from other NPC problems. However no reductions are invoked anywhere *from* SUBGRAPH ISOMORPHISM to any other problem. Conversely, HAMILTONIAN CIRCUIT and CLIQUE are mentioned as special cases of SUBGRAPH ISOMORPHISM. This leads one to suspect by analogy that G-EMBED may not prove helpful in establishing Σ_2^p-completeness for open candidates for this class. See [Pap95, §17.3] for some examples. In other words, one might conjecture that very general embedding problems are high in the hierarchy of complexity problems with respect to easy problem transformation.

The remainder of this chapter summarises and rationalises some of the results of Chapters 6 and 7.

8.6 The one formula case

Table 8.1 summarises the possibilities for permutation existence problems for wff $\phi(\vec{x})$ and a coset $C = Gk$ with $G \leq W(\vec{x})$ and $k \in W(\vec{x})$. There appear to be five cases: trivial; co-NPC; the '+' case; the C-INVAR-equivalent cases; Σ_2^p-complete. This brings together in a homogeneous format elements from the previous two chapters. Although the '+' case is certainly easier than G-EQUIV with respect to the transformation

$$(\exists g \in Gk) \models \phi^g + \phi \quad \Leftrightarrow \quad (\exists j \in G) \models \phi^j \leftrightarrow (\neg\phi)^{k^{-1}}$$

and therefore easier than C-INVAR by the equivalence of this problem and G-EQUIV (Theorem 6.3.1), we do not know if it reduces in the other direction as well.

8.7 Unresolved cases

We summarise some unresolved cases for the problems of Chapters 6 and 7. Formula types presenting problems are the Horn formulas with unbounded clause size and 2-

Problem	Complexity	Justification
$(\exists g \in C) \models \phi^g \wedge \phi$	co-NPC	equivalent to $\models \phi$
$(\exists g \in C) \models \phi^g \wedge \neg\phi$	–	always false
$(\exists g \in C) \models \neg\phi^g \wedge \phi$	–	always false
$(\exists g \in C) \models \phi^g + \phi$	C-INVAR-easy	instance of G-EQUIV
$(\exists g \in C) \models \phi^g \vee \phi$	Σ_2^p-complete	Theorem 7.5.1
$(\exists g \in C) \models \neg\phi^g \wedge \neg\phi$	co-NPC	equivalent to $\models \neg\phi$
$(\exists g \in C) \models \phi^g \leftrightarrow \phi$	C-INVAR	–
$(\exists g \in C) \models \phi^g \rightarrow \phi$	C-INVAR	Lemma 2.9.1
$(\exists g \in C) \models \phi \rightarrow \phi^g$	C-INVAR	Lemma 2.9.1
$(\exists g \in C) \models \neg\phi^g \vee \neg\phi$	Σ_2^p-complete	Theorem 7.5.1

Table 8.1: Classification of the permutation existence problem for one formula.

DNF formulas in certain cases.

Although tests of the form $\models \phi^g \leftrightarrow \psi$ are polynomial for pairs of Horn formulas which means that the problem G-EQUIV is in NP for Horn formulas, no transformation of the restricted problem *to* OPS is apparent. Transformation *from* OPS is clear by (i) of Theorem 6.2.1. We showed other tractable formula types such as k-Horn or 2-CNF were reducible to OPS by Theorem 6.2.2. There are two possibilities. Either the Horn case is equivalent to OPS by some construction unknown to us, or it is a strictly harder problem, which seems interesting and plausible in view of Conjecture 8.2.1 that the unbounded CNF case constitutes the hardest problem for G-EQUIV. Recall that in the k-CNF case the problem is in $\mathrm{P}^{\mathrm{NP}[\log n]}$. In the former case it was indicated in §8.2 that the unboundedness seems to be the source of the intractability. We have also failed to isolate the complexity of G-ISF for Horn formulas. We showed in Theorem 7.3.2 that it was NPC for k-CNF, polynomial for k-Horn or 2-CNF. It is not apparent that it is in NP for unbounded CNF formulas even if they are in Horn form.

An interesting case where 2-DNF seems problematic is again in the G-ISF problem. It was shown to be Σ_2^p-complete for 3-DNF. The 2-DNF case is not obviously in NP. Less interesting was the case for G-EMBED involving one formula in 1-CNF the other

in 2-DNF. This case is in NP but it is not obviously either NPC or polynomial.

GRAPH ISOMORPHISM was shown reducible to the G-equivalence problem for a pair of OBDDs and that the latter problem is in NP (Theorem 6.6.1). This seems to to be a reasonable candidate for an equivalence problem in NP being shown to be NPC. It seems like a very tough problem to show in general that G-equivalence for a pair of OBDDs can be polynomially reduced *to* GRAPH ISOMORPHISM or even the slightly harder OPS. The two OBDDs, based on different orderings, may have completely different graph structure even though they represent structurally identical functions.

8.8 Summary

We have presented a number of summarising remarks and conjectures concerning the results of Chapters 6 and 7. These remarks are intended to conclude the main points raised by the thesis. However in the next chapter we pursue the links between group action and k-CNF formulas from a different angle, motivated by the indications discussed in this chapter that there is a subtle element of tractability inherent in these restricted formulas that requires further exploration.

Chapter 9

Horn-Closure

9.1 Introduction

This chapter examines some properties and experimental results concerning a method of computing an approximation to $\mathsf{Close}(\phi) \stackrel{\text{def}}{=} J_{3,C,\mathsf{Vars}(\phi)}(\phi)$ for a 3-CNF formula ϕ which exploits the polynomial complexity of computing $\mathsf{Close}(\psi)$ for a 3-Horn formula ψ established in Theorem 5.5.1. A 3-CNF formula $\phi(\vec{x})$ has the nice property that it can be viewed as consisting of a Horn formula ψ and the image θ^f of a Horn formula θ under the map $f \in \Delta(\vec{x})$ which flips the sign of each variable. Applying closure to these two components and taking the conjunction, 'Horn-closure', may permit the extraction of further symmetries of the original formula which were not available syntactically. We also show how the process can be iterated. In certain cases this can result in all, or nearly all, of $\mathsf{Close}(\phi)$ being generated within a reasonable number of iterations and where each iteration runs in polynomial time. We discuss experimental results for random 3-CNF formulas and for 3-CNF encodings of the pigeonhole problem. We outline some relationships between symmetry and 'Horn-maximal' formulas, those which cannot be further Horn-closed.

We bring together several aspects from previous chapters. An application of the Horn-closure method provides an additional means of extracting non-syntactic symmetries of a formula with reasonable cost, further to the techniques mentioned in Example 3.5.2 and §4.7.1. Symmetry arguments concerning the procedure provide further illustrations

of the general principles at work in Chapter 4. The underlying motivation for studying the properties of the procedure was to explore from a different angle the relationship between group action and k-CNF formulas, especially in regard to the fact, as evidenced by the results of Chapters 6 and 7, that questions of equivalence and containment with respect to groups and computation of semantic symmetries for these restricted classes seem to admit lower complexity bounds than for arbitrary formulas, or even CNFs of unbounded clause length.

9.1.1 Plan of chapter

In §9.2 we define the Horn-closure process and review the complexity considerations. A symmetry argument is given with an example to show how this process can lead to extraction of symmetries of a wff which are not available syntactically. We also discuss why, in the n-queens problem and pigeonhole problems, this process does not help to find any more symmetries. Horn-closure can be iterated by applying permutations to the formulas, thereby changing the Horn/non-Horn structure of the clauses. In §9.4 we describe a randomised procedure that exploits the clauses added in previous iterations and which attempts to compute as much as possible of $\mathsf{Close}(\phi)$. The procedure was tested on 3-CNF versions of the pigeonhole problems. These formulas are unsatisfiable and so the closure consist of all 3-clauses over the variables, a large number, but still polynomially bounded in the input size. For $n \leq 6$ pigeons the procedure converges to the set of all clauses. For numbers greater than this there seems to be a well-defined point, quickly arrived at, where iterated Horn-closure fails to add any further clauses. This leads to the concept of a Horn-maximal formula, one which is Horn-closed under any permutation. In §9.3 we give a argument for exploiting symmetry in the determination of whether a formula is Horn-maximal, a problem that appears to be hard, and we outline some complexity considerations for these formulas.

The problematic formulas arrived at in the pigeonhole experiments are very large and prohibit combinatorial analysis such as extracting syntactic symmetries. In order to search for smaller candidate Horn-maximal formulas, some experiments were conducted

on random 3-CNFs using the criterion of achieving 7/8 of the size of $\mathsf{Close}(\phi)$ as a measure of the success of iterated Horn-closure (we explain why this criterion is used). In §9.5 we describe these experiments. It is observed that candidate Horn-maximal formulas do not arise from random formulas with less than about 40 variables, and this seems to corroborate the pigeonhole results in the sense that the pigeonhole formulas with less than 43 variables showed convergent behaviour under iterated Horn-closure. A further observation is that, in the tested range, the problems that iterated Horn-closure finds hardest are in the under-constrained region for the random formula model. We ran tests for 10 to 70 variables on 'hard' random formulas and found that iterated Horn-closure is successful in converging to 7/8 of the size of $\mathsf{Close}(\phi)$ for formulas in this range. We summarise these results and the rest of the chapter in §9.6.

9.2 Horn-closure

A 3-CNF ϕ formula can be viewed as the conjunction $\phi = \psi \wedge \theta$ of a 3-Horn formula ψ consisting of all the Horn clauses of ψ, and the remainder θ which is also essentially (for complexity purposes) a 3-Horn formula, we simply swap the polarity of every literal (apply the element f of the centre of W) to obtain a 3-Horn formula. This chapter focuses on closures of 3-CNFs and so we adopt the following notation to abbreviate the general closure operator of §3.2.1:

$$\mathsf{Close}(\phi) \stackrel{\text{def}}{=} J_{3,C,\mathsf{Vars}(\phi)}(\phi).$$

Define the Horn-closure $\mathsf{HClose}(\phi)$ as the conjunction of $\mathsf{Close}(\psi)$ and $\mathsf{Close}(\theta)$. It is clear that $\mathsf{Close}(\phi) \models \mathsf{HClose}(\phi) \models \phi$ since if $\psi \models c$ or $\theta \models c$ then $\phi \models c$. Therefore we have that ϕ is logically equivalent to $\mathsf{HClose}(\phi)$. The size of $|\mathsf{Close}(\phi)| \leq |\mathsf{Cl}_3(\mathsf{Vars}(\phi))|$ is bounded by a polynomial in $n = |\mathsf{Vars}(\phi)|$, namely $2^3.\binom{n}{3}$ or $\frac{8}{6}(n^3 - 3n^2 + 2n)$ which is $\Theta(n^3)$. Any special classes of formulas where determining $\phi \models c$ for 3-clause c is provably polynomial will therefore give an overall polynomial algorithm for computing $\mathsf{Close}(\phi)$ (Theorem 5.5.1). Hence

Lemma 9.2.1 There is a polynomial algorithm for computing $\mathsf{HClose}(\phi)$.

When ψ is 3-Horn, then $\psi \wedge \neg l_1 \wedge \neg l_2 \wedge \neg l_3$ is a Horn formula (not exactly a 3-Horn formula) and testing for satisfiability is therefore polynomial (e.g. [DG84]). Note we add clause c to the closure of ψ when $\psi \wedge \neg c$ is unsatisfiable. The following lemma helps to show why Horn-closure need only be applied once, indeed why it is justified to call it a 'closure'.

Lemma 9.2.2 If ψ is a 3-Horn formula then so is $\mathsf{Close}(\psi)$.

The proof is by considering the four different sign possibilities for a clause c: no negative signs up to all three negative signs. If $c = l_1 \wedge l_2 \wedge l_3$ is a non-Horn case then it is straightforward to check that evaluation of the units in $\psi \wedge \neg l_1 \wedge \neg l_2 \wedge \neg l_3$ leads to a Horn formula with no positive singletons. It must therefore be satisfied by the model which sets every variable false. Therefore there are no non-Horn 3-clauses c such that $\psi \wedge \neg c$ is unsatisfiable and hence $\mathsf{Close}(\psi)$ must consist only of Horn 3-clauses.

Corollary 9.2.3 $\mathsf{HClose}(\mathsf{HClose}(\phi)) = \mathsf{HClose}(\phi)$.

9.2.1 An application to finding symmetry

We have met two ways which can find a larger subgroup of $\Sigma(\phi)$ than is possible by reduction to graph automorphism using the method of §3.5.1. One depends on the representability (§2.9.2) of some known group of symmetries and the ability in certain cases to infer a larger group because all boolean functions with the smaller group must also have the larger group. The other was the method of Boy de la Tour (§4.7.1) where combining the known group with properties of binary clauses can yield further symmetries. Here we look at a method which can in theory obtain more symmetries by applying Horn-closure and then reducing to graph automorphism. Since Horn-closure is polynomial (Lemma 9.2.1) then this constitutes a polynomial reduction to graph automorphism, although it is not guaranteed that more symmetries will be found if there are some. Recall that the general case for finding semantic symmetries of a formula seems to be much harder (Theorems 5.2.1 and 6.5.1).

The following lemma relates the symmetries of a formula ϕ consisting of two components ψ and θ:

Lemma 9.2.4 Let formula $\phi = \psi \wedge \theta$. Then $\Sigma(\psi) \cap \Sigma(\theta) \leq \Sigma(\phi)$.

Proof If $g \in \Sigma(\psi)$ and $g \in \Sigma(\theta)$ then $\psi^g \equiv \psi$ and $\theta^g \equiv \theta$ and so $\psi \wedge \theta \equiv \psi^g \wedge \theta^g \equiv (\psi \wedge \theta)^g$. Hence $g \in \Sigma(\psi \wedge \theta) = \Sigma(\phi)$. □.

The following example illustrates how this works.

Example 9.2.5 Let α be the clause system below:

$$\{\{\neg x, y, z\}, \{x, \neg y, z\}, \{x, y, \neg z\}, \{\neg x, z, w\}, \{x, \neg z, w\}, \{y, \neg z, w\}, \{\neg x, \neg y, z\},$$

$$\{\neg x, y, \neg z\}, \{x, \neg y, \neg z\}, \{x, \neg y, \neg w\}, \{\neg x, z, \neg w\}, \{x, \neg z, \neg w\}\}.$$

The formula has a syntactic group G of order 2 generated by $(x\ \neg z)(\neg x\ z)(y \neg y)(w \neg w)$, but $|\Sigma(\alpha^C)| = 24$. It splits into Horn and non-Horn parts each of size 6 which have closures of size 9. The closures each have the same syntactic group (and therefore semantic group, by Lemma 3.2.3) G' of order 6 and therefore by Lemma 9.2.4 we know that the intersection G', which is strictly larger than the above syntactic group, is a subgroup of $\Sigma(\alpha^C)$.□

Pigeonhole and n-queens problems The usual encodings of the pigeonhole and n-queens problems include a large number of binary constraints and we know from Theorem 5.5.1 that computing closures of 2-CNF formulas is also polynomial. The formulas were analysed to see if application of closure to the 2-CNF part would yield a larger group of symmetries when intersected with the syntactic symmetries of the remaining clauses. In the queens case, all the binary clauses contain two negated variables. One can quickly check that this means that the 2-CNF part of the queens problems are already closed and so no more symmetries can be extracted. Note that in the limiting case, the queens problems probably have no more than 8 semantic symmetries, though for small n they can have more as a result of having only a few solutions.

In the pigeonhole case the 2-CNF part is also closed. The structure of the symmetry groups of each part is interesting. For n pigeons, the binary clauses have syntactic group $S_n \wr S_{n-1}$ and the remaining clauses have group $S_{n-1} \wr S_n$ (see §2.5 for wreath products). The intersection of these groups is the standard syntactic group for this problem which corresponds to the direct product $S_n \times S_{n-1}$. In §9.4 below we look at a way of applying Horn-closure to the pigeonhole problem, by turning it first into a 3-CNF.

It is quite easy to construct formulas such as that of Example 9.2.5 in which the technique works, however it is perhaps of not much use in practice. In the next section we look at how knowledge of symmetry in ϕ can assist computing $\mathsf{Close}(\phi)$ and $\mathsf{HClose}(\phi)$.

9.2.2 Utilising symmetry

If $G \leq \Sigma(\phi)$ then $\mathsf{Close}(\phi)$ consists of the conjunction of whole orbits of G acting on the set of all clauses over the variables of ϕ. Therefore for any clause c, if $\phi \models c$ then the whole orbit of c under G can be added to ϕ. Thus using the naive method of testing each clause for inclusion (which might be effectively optimal) in $\mathsf{Close}(\phi)$, symmetry has considerable benefits, since it reduces to testing just one member of each G-class of the clauses. This also applies to the separate parts in Horn-closure by:

Lemma 9.2.6 Let $G \leq \Sigma(\phi)$. If ψ is the Horn part of ϕ and $\psi \models c$ then the Horn part of the orbit of c under G is a subset of $\mathsf{Close}(\psi)$.

Proof If $\psi \models c$ then $\phi \models c$ hence the orbit of c under G is a subset of $\mathsf{Close}(\phi)$. Therefore the Horn part of the orbit is a subset of $\mathsf{Close}(\psi)$. □

Therefore, after splitting into Horn and non-Horn parts, we still have the benefit of the symmetry in terms of adding whole orbits of clauses if just one member is found to be in the closure of one of the parts.

Note The orbit of a clause under $G \leq W(\vec{x})$ may contain both Horn and non-Horn clauses in the case that G contains elements that affect complementarity. If $G \leq S(\vec{x})$

then the orbit of c would be either all Horn or all non-Horn.

In Example 9.2.5 it has been checked that the syntactic group of the original formula partitions the sets of all Horn and non-Horn clauses into orbits of size 2 and so applying the symmetry argument in this example would halve the amount of clauses that would have be tested for in each closure.

Looking ahead We see in §9.4 that one application of Horn-closure is not the end of the story, even though we view it as a closure operator by Corollary 9.2.3. Horn-closure is certainly not sufficient in general on its own to generate $\mathsf{Close}(\phi)$ but we investigate the question of whether a randomised iterated version can do this. For instance, in the formula of Example 9.2.5 the full 18 clauses are obtained within 2–4 steps of iterating the procedure. This then enables us to compute all 24 semantic symmetries by reduction to graph automorphism. We first examine a property of formulas that would cause the proposed iterated Horn-closure algorithm to fail.

9.3 Horn-maximality

Call a 3-CNF formula $\phi(\vec{x})$ 'Horn-maximal' if for every permutation $\delta \in \Delta(\vec{x})$ and split of ϕ^δ into Horn and non-Horn parts ψ and θ, then ψ and θ are closed. If ϕ is closed then ϕ is Horn-maximal: this is a corollary of Lemma 9.2.2 that only Horn clauses can be added to a Horn formula under closure, and dually. The problem of Horn-maximal formulas which are not closed is one which features in the rest of this chapter. We discuss here aspects relating Horn-maximality with symmetry and complexity.

Symmetry Knowing a syntactic symmetry group $G \leq S(\vec{x})$ of formula $\phi(\vec{x})$ assists in determining Horn-maximality since the splitting permutations $\Delta(\vec{x})$ are partitioned by the group G into classes that give equivalent behaviour for Horn-closure.

Lemma 9.3.1 Let $G \leq S(\vec{x})$ be a syntactic group of ϕ. If all representatives δ of the G-classes on $\Delta(\vec{x})$ are such that $\mathsf{HClose}(\phi^\delta) = \phi^\delta$ then ϕ is Horn-maximal.

Proof We need to show for all $g \in G$ that if ϕ^δ is Horn-closed then so is $\phi^{g^{-1}\delta g}$. Since g^{-1} is a symmetry of ϕ we see that this asks if $(\phi^\delta)^g$ is Horn-closed iff ϕ^δ is, which follows since $g \in S(\vec{x})$ does not affect the distribution of Horn-clauses.□

For example, if formula ϕ is syntactically invariant under $S(\vec{x})$ then only $|\vec{x}| + 1$ tests would be required to establish Horn-maximality: just test the δ with different numbers of moved points. In fact with symmetry we have double benefit in this problem because the symmetry also simplifies the question of closure by Lemma 9.2.6 for each of the representative δ.

Complexity The main observation is that in iterated Horn-closure it must be the case, in some problem instances, that an intermediate formula ϕ arises in which it is intractable to find a permutation δ such that $\mathsf{HClose}(\phi^\delta)$ is larger than ϕ. That is, unless iterated Horn-closure were to provide a tractable (probabilistic) algorithm for satisfiability. Although adding a few clauses does not help to determine satisfiability, the ability to eventually compute, by iterating a sufficient number of times, the cardinality of $\mathsf{Close}(\phi)$ does. To exhibit a polynomial algorithm for satisfiability, it would be sufficient to exhibit a polynomial-time procedure which is guaranteed to add a clause to a 3-CNF formula if the formula is not already closed. This is because there are only $\Theta(n^3)$ possible clauses to add, i.e. only a polynomial number of steps required.

Horn-maximal formulas which are not closed are not actually predicted by this argument because formulas admitting an exponentially small proportion of favourable splitting permutations would be sufficient to cause intractability. Determining whether a formula actually is Horn-maximal is a problem easily seen to be in co-NP:

Theorem 9.3.2 The language of Horn-maximal formulas is in co-NP.

Proof A polynomial-sized certificate that a formula ϕ is not Horn-maximal consists of a permutation δ, a clause c not in the Horn part ψ of ϕ^δ and a polynomial-size certificate that $\psi \models c$.□

The following section describes the iterated Horn-closure procedure and discusses experiments on the pigeonhole problem that produce strong evidence that Horn-maximal formulas do arise in this procedure.

9.4 Iterated Horn-closure

Horn-closure can be transformed into a randomised iterated procedure by the expedient of applying an element of $\Delta(\vec{x})$ to the formula at each step, that is before it is split into Horn and non-Horn parts. If a random permutation of variable signs is applied to ϕ (i.e. with probability half a variable x is replaced systematically by $\neg x$) then we obtain essentially the same formula in terms of its model structure, however the way it splits into Horn and non-Horn parts will in general be essentially different. The procedure which iterates Horn closure n times for formula $\phi(\vec{x})$ is then the following algorithm:

$$
\begin{aligned}
\textbf{for} \quad & i = 1 \text{ to } n \textbf{ do} \\
& \delta \leftarrow \text{random element of } \Delta(\vec{x}) \\
& \psi \leftarrow \text{Horn part of } \phi^{\delta} \\
& \theta \leftarrow \text{non-Horn part of } \phi^{\delta} \\
& \phi \leftarrow (\mathsf{Close}(\psi) \wedge \mathsf{Close}(\theta))^{\delta^{-1}}
\end{aligned}
$$

We look at how the procedure works for 3-CNF encodings of the pigeonhole problems.

9.4.1 Pigeonhole problems

The pigeonhole problems provide good examples of formulas which are provably hard for uniform search procedures e.g. [Hak85]. Iterated Horn-closure was tested on 3-CNF versions of these problems. The 3-CNF encoding process involves the use of auxiliary variables and produces a formula where there is a one-to-one mapping back to the models of the original formula, as discussed in §8.2. Hence unsatisfiability is preserved and one can extract models of the original from models of the 3-CNF version. Table 9.1 gives some statistics for the progress of iterated Horn-closure on pigeonhole problems for n up to 9.

n	Vars	Iterations	Initial	Final	Percent
3	7	4,5,5,5,4	13	280	100
4	13	5,6,6,6,7	26	2,280	100
5	26	9,8,9,9,9	64	20,800	100
6	43	15,19,16,14,15	127	98,728	100
7	64	16,17,17,19,18	221	33,908*	9.8
8	89	13,13,13,12,12	352	76,772*	11.8
9	118	13	526	155,001*	13.8

Table 9.1: Statistics for iterated Horn closure on pigeon hole problems. Note that the number of variables refers to the 3-CNF version. The starred values were run to a total of 30 iterations: the numbers in the Iterations column indicate the stage at which they peaked at the final value.

The important point is that these formulas proved problematic for iterated Horn-closure. For 7 pigeons there is a relatively rapid rise to a formula consisting of 33,908 clauses. The same formula (we refer to it now as P7) seems to be arrived at every time, that is, it is not a permutational variant nor some completely different formula of the same number of clauses. The resulting formula does not contain a set of eight clauses consisting of all possible sign variants on three variables, so it is not possible to say immediately that it is unsatisfiable. (In fact experimentation showed that if a formula does contain such a trivially unsatisfiable formula, it converges to all clauses in about three iterations.)

Note We estimate that about 2/3 of the clauses in the incomplete formulas can be accounted for as extensions of both original binary clauses and the binary clauses introduced in the 3-CNF construction, all of which are padded with a dummy literal forced to take value 0.

The incomplete formulas seem to be good candidates for being Horn-maximal. What is interesting is that the pigeonhole problems for less than 7 pigeons converge. We see later in §9.5 that there is further evidence that candidate Horn-maximal formulas are only start appearing beyond 40 or so variables. We continue this section with

discussion relating to the Horn-maximality, or otherwise, of these formulas.

Unequal splits Can some heuristic applied to choice of splitting permutation lead to better overall performance of iterated Horn-closure? Certainly there are reasonable methods for obtaining an *unequal* split. The idea behind an unequal split is that the larger formula will be more constrained and its closure corresponding greater than formulas of about half the size of the input formula. A greedy method has been implemented which attempts to maximise the number of Horn clauses by iteratively flipping variable signs. The process is terminating and of polynomial complexity. $|\mathsf{Vars}(\phi)|$ flips are sufficient from any random starting permutation: it always becomes the case that there is no further way of increasing the number of Horn clauses by flipping variables. A split of about $1:2$ seems to be always possible with a set of random clauses, and in general we can obtain quite unequal splits with a small amount of work.

In the P7 formula arising from the 7 pigeon problem this algorithm produced a split of 30,583 : 3,325 with 36 flips. However this partition still failed to add any clauses. The formula is therefore stable under extreme permutations which would be very rare in a random distribution. One point is the success of the heuristic in producing an unequal split. By comparison with random sets of clauses, the fact that a very unequal split is produced in this formula indicates some structure that is linked with its method of construction. (Random permutations produce equal splits with the formula.) We will discuss this heuristic further in the summary of this chapter.

Symmetry As we saw in §8.2 group action is not well-preserved by the conversion to 3-CNF: if we could decide G-embeddings of CNFs by reducing them to 3-CNFs this would entail $\Sigma_2^p \subseteq \mathrm{P}^{\mathrm{NP}[\log n]}$. We could not expect symmetry to be preserved by the same token. However the pigeonhole problems are unsatisfiable and so they are still semantically invariant under all permutations after 3-CNF encoding. For unsatisfiable formulas we might then expect some symmetry properties in a Horn-maximal representation which is not closed: it is a symmetrical concept in the sense of implying an 'invariance' over all possible permutations of variable signs.

A non-empty, non-closed formula $\phi(\vec{x})$ cannot be itself syntactically invariant under

$\Delta(\vec{x})$ unless it contains a trivially unsatisfiable subformula, i.e. the orbit of a clause under $\Delta(\vec{x})$. We already noted the formula P7 does not contain such a formula and therefore it does not have this symmetry, even though the underlying function does. The P7 formula does not exhibit full syntactic symmetry of variables (or any conjugate group) either. It is not big enough. The smallest fully symmetric formula of 64 variables has size $\binom{64}{3} = 41{,}664$ since this is the size of the orbit of a single 3-clause under the group S_{64}.

An attempt was made to discover the syntactic group of the set of clauses using the graph construction of §3.5.1 and the *nauty* graph automorphism package of [McK92]. However, the structure seems to be prohibitively large. We showed in §9.3 that if we did find symmetry in P7, this would then assist in determining Horn-maximality.

9.5 Horn-maximality and random 3-CNFs

In this section we describe some experiments aimed at locating Horn-maximal candidates among random 3-CNF formulas. We have already seen two behaviours for iterated Horn-closure, i.e. in the pigeonhole problems of §9.4.1: convergence to $\mathsf{Close}(\phi)$ and convergence to some formula considerably smaller than $\mathsf{Close}(\phi)$. In the second case the formulas seemed highly stable and possibly Horn-maximal. Broadly, two further behaviours have been noted with iterated Horn-closure of random 3-CNFs and we propose a measure of success of iterated Horn-closure that helps to delineate the various behaviours. First we observe:

Lemma 9.5.1 It is sufficient that 3-CNF ϕ contains more than $\frac{7}{8}|\mathsf{Cl}_3(\mathsf{Vars}(\phi))|$ clauses for it to be unsatisfiable.

If ϕ contains more than the stated number of clauses then it must contain a subset of size eight consisting of all sign variants on three variables, i.e. it must contain an unsatisfiable subformula. At the outer extreme of the range of satisfiable formulas are those with just one model, these have closures with exactly the above critical number of clauses. We propose the 7/8 criterion as a measure of the success of iterated Horn-

closure: that is we want to know how many iterations are required to reach this fraction of the size of Close(ϕ) in the general case. It is a measure appropriate to analysis of randomised procedures, with the added property of deciding satisfiability in the case that the formula increases beyond 7/8 of all clauses. We broadly observe the following behaviours:

i). The formula grows quickly and converges to exactly Close(ϕ). These are less common and are usually unsatisfiable examples.

ii). The formula grows quickly and converges to within a very small fraction of the size of Close(ϕ) and then slows, with extra clauses being increasingly difficult to add.

iii). The formula grows slowly, perhaps looping for a few iterations here and there, but eventually reaches 7/8 of the size of Close(ϕ).

iv). The formula loops at some fraction often considerably less than 7/8 and resists many further iterations. Trivially satisfiable formulas are often the culprits here but hard unsatisfiable problems have been found also to exhibit this behaviour as in the pigeonhole problems.

The 7/8 criterion is thus designed to differentiate between case iv) problems with a tendency to arrive at non-closed Horn-maximal formulas and cases i)–iii) those which either converge quickly or which give the appearance that after a sufficient number of iterations, they would eventually do so.

Random 3-CNF We use the method of generating random 3-CNFs of [MSL92]: a set of three variables is chosen at random for each clause, and variables are then negated with probability 1/2. We shall refer to the 'hardness' of the formulas in terms of various criteria established by well-known experimental work. Problems are divided into three ranges: the under-constrained range where nearly all formulas are satisfiable, the over-constrained range where nearly all formulas are unsatisfiable and the critically-constrained range in between. The 'cross-over' point, the number of

clauses c where about half of random 3-CNF formulas of v variables are satisfiable was shown experimentally to coincide very closely with the hardest formulas in terms of resources required to establish satisfiability [MSL92]. With the above model of random formula, it was shown that the hard formulas require time that appears to grow exponentially for all tested satisfiability algorithms [Mit93]. This cross-over point seems to be related almost, but not quite, linearly with the number of variables in the formula. A close fit, for 20–300 variables, to the observed relationship between the numbers of variables and clauses at the cross-over point has been given by the formula $c = 4.258v + 58.26v^{-2/3}$ in [CA96].

9.5.1 Preview of experiments

Experiment 1 Iterated Horn-closure was tested extensively on random 3-CNF formulas with up to 30 variables in an attempt to find out whether candidate non-closed Horn-maximal formulas occurred with any frequency in reasonably simple problems. For $v = 10, 20, 30$ variables the procedure was tested on problems ranging from size v to $8v$, recording mean number of iterations, over 200 formulas, that were required to reach 7/8 of closure size.

Experiment 2 At 40 variables some highly resistant formulas appear, even though most of the seemingly difficult ones will eventually converge. Iterated Horn-closure was tested on formulas of 80–160 clauses, recording mean percentage, over 50 formulas, of the size of Close(ϕ) obtained after 10, 20 and 30 iterations

Experiment 3 Iterated Horn-closure was tested on 'hard' problems of $4.5v$ clauses for $v = 10, 15, \ldots, 70$ recording mean number of iterations, over 50 formulas, required to reach 7/8 of Close(ϕ) (which always occurred).

9.5.2 Up to 30 variables

For up to 30 variables we see fully convergent behaviour for iterated Horn-closure on random formulas, using the 7/8 criterion (Lemma 9.5.1 and surrounding discussion). That is, on all formulas tested, a sufficient number of iterations will cause the formula

ϕ to increase in size until it is at least 7/8 of the size of Close(ϕ). Figure 9.1 expresses these results for 10, 20 and 30 variables using the horizontal axis to describe the number of clauses in the formulas as a multiple of the respective number of variables. Iterated

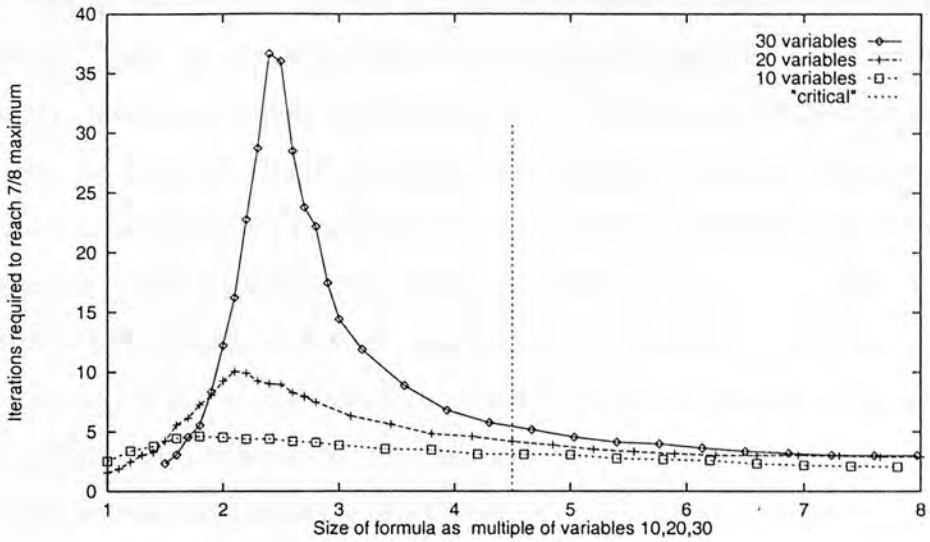


Figure 9.1: The number of iterations of Horn closure required to reach 7/8 of total possible for random formulas of 10, 20 and 30 variables and v to $8v$ clauses.

Horn-closure performs poorly on under-constrained formulas. On the other hand, in the 30 variable case, it will converge to a mean size of about 30,000 clauses, where ϕ contains initially 135 clauses, in about 5.5 iterations on average. Therefore the first point is that hardness for iterated Horn-closure does not coincide with the cross-over point for random 3-CNFs in this range. However the second point is that the peak difficulty in terms of multiples of variables has an apparent rightwards momentum. The peak difficulty seems to be quite clearly defined for 10, 20 and 30 variables at respectively $1.8v$, $2.1v$ and $2.4v$ clauses.

Conclusion Random formulas with up to 30 variables do not appear to give rise to strong candidates for non-closed Horn-maximal formulas.

We next look at how the procedure performs at 40 variables, and return to the question of how iterated Horn-closure performs on the harder formulas in the following section.

9.5.3 40 variables

At 40 variables and formulas ϕ of about $110 = 2.75v$ problems are encountered on which iterated Horn-closure appears to loop at less than the required 7/8 of the size of Close(ϕ). These are the first candidates for being Horn-maximal that have so far appeared in these experiments. Of the formulas at $2.75v$ clauses that grow very slowly, most seem to eventually converge after a few hundred iterations, where frequently several iterations are required to add a new clause. It seems as though the distribution of favourable splitting permutations in these formulas is low but non-zero. However one formula has certainly been found that resists 1,000 iterations. This formula has a closure size of 213 and reaches a formula of size 172 after 13 iterations, after which no further additions could be made. The problem of determining whether this formula really is Horn-maximal presents severe difficulties. It is satisfiable and almost certainly has no symmetry (see §2.9.3) therefore we cannot utilise Lemma 9.3.1. Even if was feasible to perform 2^{40} tests of any description, the fact that Horn-closure has the underlying $\Theta(n^3)$ complexity aspect, in terms of the number of possible candidate clauses that could be added, completely prohibits any brute force approach on the lines of testing that every permutation δ leads to a Horn-closed formula.

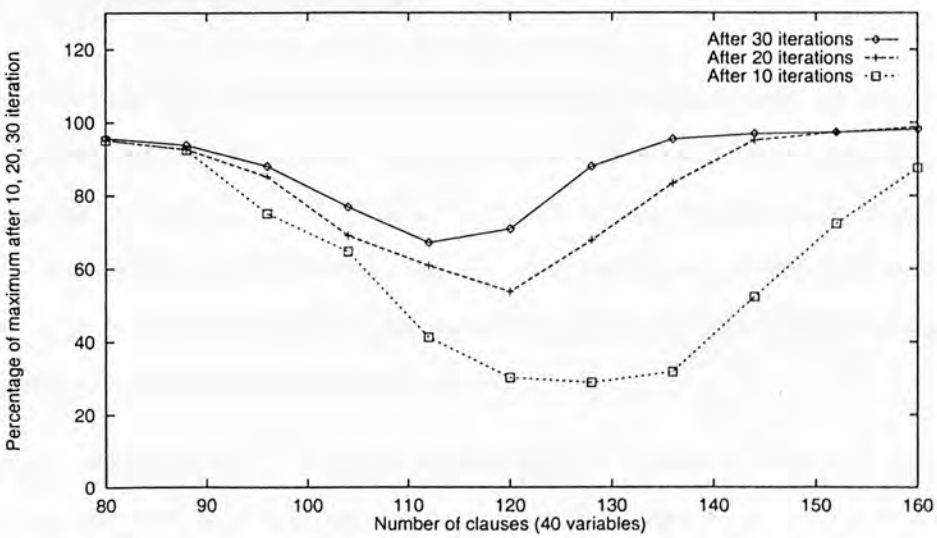


Figure 9.2: The percentage of maximum closure size obtained by iterated Horn-close for 10, 20 and 30 iterations on random formulas of 40 variables and 80–160 clauses.

The graph of Figure 9.2 shows how the percentage of Close(ϕ) obtained varies with the number of iterations. The graphs more accurately reflect the point of maximum hardness for iterated Horn-closure as the number of iterations is increased. The reason for the changing position of the graphs is connected with the size of closures towards which iterated Horn-closure attempts to converge. As the size of the input formula increases, the formulas are more constrained and have larger closures. A small fixed number of iterations will generate initially a larger proportion of the closure on the smaller formulas. However as the iterations are increased, the problems in the hard area slow down while the formulas with larger closures begin to converge.

Conclusion We have established that strong candidates for Horn-maximal formulas which are not closed can be derived by applying iterated Horn-closure to random 3-CNFs of 40 variables and 110 clauses.

At $v = 40$ variables we still have 7/8 convergence with high probability for formulas of more than about $3v$ clauses. At $4.5v$ clauses convergence always occurs, on average after 6.7 iterations (Figure 9.3). In the next section we try to determine how effectively iterated Horn-closure works on random formulas of $4.5v$ clauses.

9.5.4 Behaviour at $4.5v$ clauses

We have seen for up to 40 variables that iterated Horn-closure finds problems in the under-constrained region hardest. This section presents results for formulas of $4.5v$ variables for $v = 10, 15, \ldots, 70$. These are marginally over-constrained formulas for $v > 40$ (with reference to the experimentally observed cross-over point), however the number $4.5v$ is maintained for simplicity and because we need a fixed measure for comparing results at different variable numbers.

Practical limitations on the implementation used to compute iterated Horn-closure mean that sample sets of any size for more than 70 variables are not feasible within the scope of this research (the implementation is described in Appendix B), however a number of tests were run at 100 variables and we discuss the observations. The performance of iterated Horn-closure on formulas in the 10 to 70 variable range is

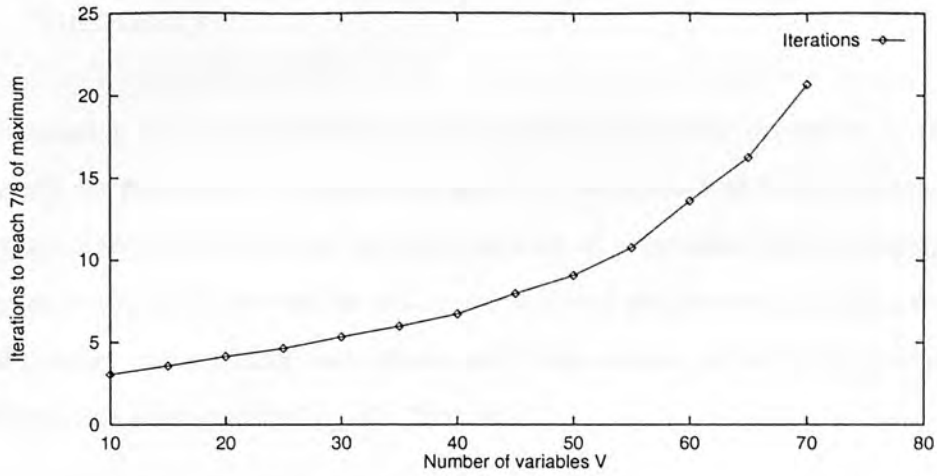


Figure 9.3: The number of iterations of Horn-closure required to reach 7/8 of total possible size starting from $4.5v$ random clauses.

surprisingly robust. The variation in the number of iterations required at 70 variables was between 12 and 36 with a mean of 20.66 over the 50 runs. Furthermore the graph of Figure 9.3 seems to promise continued smooth behaviour.

Evidence, however, that the hard region for iterated Horn-closure, seen moving rightwards in Figure 9.1, moves eventually towards the harder formulas, comes from a number of runs at 100 variables and 450 clauses. The behaviour changes from rapid exponential-like growth at 70 variables, to sporadic growth with periods of no growth at all at 100 variables, similar to what was observed at $2.75v$ clauses at 40 variables. The three formulas, one satisfiable, which were tested did in fact converge to 7/8 of a mean of about 1 million clauses after 212, 268 and 701 iterations, with this last run requiring 67 hours of CPU time.

Conclusion Though well behaved up to 70 variables, iterated Horn-closure on the hardest random formulas is predicted by these experiments to meet with non-convergent behaviour at more than 100 variables.

9.6 Summary

In this chapter we have examined some properties of a closure procedure on 3-CNFs which can be performed in polynomial time. It was shown that application of the procedure could lead to finding more symmetries of a formula, with reasonable cost compared to the worst case results of Chapter 6. It was also shown that known symmetries of ϕ assist in computing both closure and Horn-closure, as well as in the problem of determining Horn-maximality of a formula.

We described a way of iterating Horn-closure which exploits clauses added in previous iterations. It was not established whether the incomplete pigeonhole formulas of §9.4.1 really are Horn-maximal or just intractable in terms of finding favourable splitting permutations. It was also not established whether these formulas have any syntactic symmetry, something that we proposed could be associated with Horn-maximal representations of unsatisfiable formulas.

As evidence of the nontriviality of the property of Horn-maximality we have noted in both the pigeonhole problems and in the experiments of §9.5 that non-closed candidates only seem to start appearing at 40 variables and when they do occur at this level it is difficult to establish Horn-maximality with brute force. We discovered a region of under-constrained random formulas with 40 variables which seemed to give rise to strong candidates for Horn-maximal formulas which are not closed. On the other hand we showed that these do not emerge from random formulas with fewer than 30 variables.

The hypothesis that iterated Horn-closure has convergent behaviour on hard random formulas seems unlikely to hold in the light of the observations of a momentum of the hard point for the procedure towards the cross-over point for random 3-CNFs. Although good behaviour was observed up to 70 variables, it was predicted that non-convergent behaviour would appear regularly at beyond 100 variables. Testing extensively at this level or beyond was not feasible with the implementation used, whose underlying algorithms are briefly described in Appendix B.

There is also the question of whether it is profitable trying to find heuristics that improve the overall growth pattern under iterated Horn-closure. One such heuristic, involving unequal splits of the formula discussed in §9.4.1, turns out to be not as useful as anticipated when iterated. The result of applying Horn-closure with an unequal split is to obtain an even more unequal split because the more constrained part will increase by more than the less constrained part. Recall Lemma 9.2.2 which points out that only Horn clauses will be added to a Horn formula, and dually. Taking the union of the new parts, then any reasonable heuristic for finding a new unequal split starting from a random permutation is quite likely to converge approximately to the previous one, where now these two parts are already closed. This is what appears to happen when this heuristic is used. The permutations tend to converge to the previous permutations and this has an overall negative effect on performance. An approach which uses the unequal split heuristic intermittently in the iterated procedure seems to give good results. This is incorportated as an option in the implementation described in Appendix B. The general impression from the experiments in this chapter is that randomness in the choice of splitting permutation is a good heuristic.

Chapter 10

Summary and Further Work

10.1 Contributions

In this final chapter we conclude by summarising the contributions of the thesis. These are divided into four areas. We first point out where a contribution has been made in terms of collating material in an area lying at the intersection of a considerable number of theoretical issues. Then some minor contributions of miscellaneous symmetry arguments and reduction tools are summarised. We then review the complexity results of this thesis that have extended or complemented results in the literature. Finally we review the justifications for the main work of the thesis in introducing and investigating in detail the equivalence and containment problems of Chapters 6 and 7 and the iterated Horn-closure procedure of Chapter 9.

Collation of material Chapters 2, 3 and 4 which present respectively background material for the thesis, a collection of relevant transformation techniques and a survey of symmetry methods, are essentially collatory in nature. We feel this is advantageous to the reader in simplifying the structure of the latter and more important half of the thesis. Collecting together the transformation techniques of Chapter 3 provides a self-contained resource that could be useful for further results in this area, whereas a disseminated approach would fail to highlight the many group theory connections between the transformation techniques and therefore the many ways in which they can be combined. The survey of symmetry methods within the unified context of the

material of Chapters 2 and 3, also provides a significant resource, as well as its main function of motivating the the study of connections between symmetry and complexity in the rest of the thesis.

Minor contributions We presented an idea for a heuristic technique in §4.8 and §4.9 for exploiting the structure of groups in two different search contexts, experimental results for which showed that the group structure could be exploited independently of the problem. Two of the tools of Chapter 3, the D-transformation and the gluing technique, are mentioned as contributions only in the sense that they involve routine notions concerning groups and formulas separately, but for which some care was need to exhibit the required properties in combination. Finally the argument of §9.2.1 concerning Horn-closure and symmetry is viewed as a minor contribution in the context of other methods for extracting further symmetries of a formula which are not available syntactically.

Complexity results We gave improved results in §5.2 and §6.5 for two problems of [BdlT96a]. We showed in §7.3 that a decision problem relating to a construction of [CGLA96] was Π_2^p-complete. In §5.5 we established some complexity results for computational aspects of the Galois connection formalisation of [Mar98].

Chapters 6 and 7 In these chapters we established a number of further complexity results, however these need to be justified in terms of why they are interesting. We have already devoted Chapter 8 to this purpose, but we summarise the main points again. One of the clearest justifications of interest here is perhaps that given by the summary of Table 8.1 which shows a uniform collection of very simple problem formulations exhibiting a range of complexities between trivial and Σ_2^p-complete, indicating that these types of problem have a rich complexity structure. A second point is that reasonably natural Σ_2^p-complete problems have been exhibited, although it is not claimed that the derivations are profound in themselves. Natural Σ_2^p-complete problems are relatively rare. The third point of interest is the puzzle of the simplified complexity aspect of the k-CNF formulas in the context of these problems, established by various theorems in the two chapters, despite the fact that the restriction to k-CNF formulas in

SAT still gives hard problems. The fourth point, based around Conjecture 8.4.1 is the observation that two of the Σ_2^p-complete problems can be viewed as strong analogies of graph-theoretic questions, but with the distinguishing property that corresponding completeness proofs appear as if they would be much easier. We view the generation of these observations and the details of the complexity results connected with them as the main contribution of the thesis.

Horn-closure The subject of this chapter seems to be a rich source of material for complexity and symmetry results. We gave a number of minor results connecting Horn-closure with symmetry. We described some experiments in which we isolated various formulas with a high degree of resistance to the iterated procedure, the existence of which is predicted by the underlying tractability of the closure.

10.2 Further work

Chapter 8 contains three conjectures that we believe worthy of further investigation. We also outlined in §8.7 most of the unresolved special cases of Chapters 6 and 7. The final section of Chapter 9 refers to problems that we did not resolve in the work of that chapter, and which seem interesting.

Bibliography

[And96] H. R. Anderson. An introduction to binary decision diagrams (course notes). Technical report, Technical University of Denmark, Department of Computer Science, 1996.

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th Ann. ACM Symp. on the Theory of Computing*, pages 421–429, 1985.

[Bab95] L. Babai. Automorphism groups, isomorphism, reconstruction. In *Handbook of Combinatorics*, pages 1447–1540. Elsevier, 1995.

[BC80] M. W. Blum and A. K. Chandra. Equivalence of free boolean graphs can be decided probabilistically in polynomial time. *Information Processing Letters*, 10:80–82, 1980.

[BdlT96a] T. Boy de la Tour. Ground resolution with group computations on semantic symmetries. In *Proceedings of the 13th International Conference on Automated Deduction (CADE-13)*, pages 478–492, 1996.

[BdlT96b] T. Boy de la Tour. On the complexity of extending ground resolution with symmetry rules. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 289–295, 1996.

[BFP88] C.A. Brown, L. Finklestein, and P.W. Purdom. Backtrack searching in the presence of symmetry. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 6th International Conference (AAECC6)*, pages 99–110. Springer-Verlag: Heidelberg, Germany, 1988.

[BG82] A. Blass and Y. Gurevich. On the unique satisfiability problem. *Information and Control*, 55:80–88, 1982.

[BH88] S. R. Buss and L. Hay. On truth-table reducibility to SAT and the difference hierarchy over NP. In *Proceedings of the Third Symposium on Structure in Complexity Theory*, pages 224–233, 1988.

[BHZ87] R. B. Boppana, J. Hastad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25:127–32, 1987.

[Bry86] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE transactions on Computers*, C-35:677–691, 1986.

[Bry92] R. E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. Technical Report CMU-CS-92-160, School of Computer Science, Carnegie Mellon University, 1992.

[BS94] B. Benhamou and L. Sais. Tractability through symmetries in propositional calculus. *Journal of Automated Reasoning*, 12:89–102, 1994.

[But91] G. Butler. *Fundamental Algorithms for Permutation Groups.* Springer-Verlag: Heidelberg, Germany, 1991.

[CA96] J.M. Crawford and L.D. Auton. Experimental results on the crossover point in random 3-sat. *Artificial Intelligence*, 81, 1996.

[CEFJ96] E. M. Clarke, R. Enders, T. Filkhorn, and S. Jha. Exploiting symmetry in temporal logic model-checking. *Formal Methods in System Design*, 9:77–104, 1996.

[CGLA96] J. Crawford, M. Ginsberg, E. Luks, and R. Amitabha. Symmetry-breaking predicates for search problems. In *Fifth International Conference on Principles of Knowledge Representation and Reasoning*, pages 148–159, 1996.

[CK91] P. Clote and E. Kranakis. Boolean functions, invariance groups and parallel complexity. *SIAM Journal on Computing Science*, 20:553–590, 1991.

[Cla92] M. Clausen. Almost all boolean functions have no linear symmetries. *Information Processing Letters*, 41:291–292, 1992.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd Ann. ACM Symp. on Theory of Computing*, pages 151–158, 1971.

[Cra92] J. Crawford. A theoretical analysis of symmetry in first order logic. In *Working Notes of the AAAI Workshop on Tractable Reasoning*, San Jose, CA, 1992.

[DG84] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 3:267–284, 1984.

[DM96] J. D. Dixon and B. Mortimer. *Permutation Groups.* Springer-Verlag: Heidelberg, Germany, 1996.

[EIS76] S. Even, A. Itai, and R. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing Science*, 5:691–703, 1976.

[FHL80] M. Furst, J. Hopcroft, and E. Luks. Polynomial time algorithms for permutation groups. In *21st IEEE Symposium on Foundations of Computer Science*, pages 36–41, 1980.

[FHS78] S. Fortune, J. Hopcroft, and E. M. Schmidt. The complexity of equivalence and containment for free single variable program schemes. In *Lecture Notes in Computer Science 62*, pages 227–240. Springer-Verlag: Heidelberg, Germany, 1978.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman, 1979.

[Gla74] J. Glaisher. On the problem of the eight queens. *Philosophical Magazine*, 48:457–467, 1874.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *Proceedings of the 17th Ann. ACM Symp. on the Theory of Computing*, pages 291–304, 1985.

[GS88] S. Goldwasser and M. Sipser. Private coins versus public coins. In *Proceed. of 18th Ann. ACM Symp. on Theory of Computing*, pages 132–147, 1988.

[Hak85] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[Har71] M. Harrison. Counting theorems and their applications to classification of switching functions. In A. Mukhopadhyay, editor, *Recent Developments in Switching Theory*, pages 85–120. Academic Press, 1971.

[Hen89] M. Hennessey. *Algebraic Theory of Processes.* Massachusetts Institute of Technology, 1989.

[Hof82] C. M. Hoffman. *Group-Theoretic Algorithms and Graph Isomorphism.* Springer-Verlag: Heidelberg, Germany, 1982.

[Joh88] D. S. Johnson. The NP-completeness column: an ongoing guide. *Journal of Algorithms*, 9:426–444, 1988.

[Joh92] D. S. Johnson. The NP-completeness column: an ongoing guide. *Journal of Algorithms*, 13:502–524, 1992.

[Kis98] A. Kisielewicz. Symmetry groups of boolean functions and constructions of permutation groups. *Journal of Algebra*, 199:379–403, 1998.

[KK71] R. A. Kowalski and D. Kuehner. Linear resolution with selection function. *Artificial Intelligence*, 2(2):27–60, 1971.

[Kri85] B. Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22:253–375, 1985.

[Lub81] A. Lubiw. Some NP-complete problems similar to graph isomorphism. *SIAM Journal on Computing Science*, 10:11–21, 1981.

[Luk82] E. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25:42–65, 1982.

[LW96] W. Ledermann and A. J. Weir. *Introduction to Group Theory.* Longman, 1996.

[Mar98] U. Martin. Notes on groups and boolean formulae. Technical report, Department of Computer Science, St. Andrews University, 1998.

[Mat79] R. A. Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8:131–132, 1979.

[McK92] B. D. McKay. *nauty* user's guide (version 1.5). Technical report, Computer Science Department, Australian National University, 1992.

[Mit93] D. Mitchell. *An Empirical Study of Random SAT.* PhD thesis, Department of Computer Science, Simon Fraser University, 1993.

[MS72] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *Proceedings of the 13th Annual Symposium on Switching and Automata Theory*, pages 125–139, Long Beach, CA, 1972.

[MSL92] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distribution of SAT problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 459–465, 1992.

[Pap95] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley: Reading, MA, 1995.

[PD94] H. A. Priestley and B. A. Davey. *Introduction to Lattices and Order.* Cambridge University Press: Cambridge, England, 1994.

[PY82] C. H. Papadimitriou and M. Yannakakis. The complexity of facets and some facets of complexity. In *Proceedings of the 24th ACM Symposium on the Theory of Computing*, pages 229–234, 1982.

[S+95] Martin Schönert et al. *GAP – Groups, Algorithms, and Programming.* Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, fifth edition, 1995.

[Sim70] C. C. Simms. Computational methods for permutation groups. In *Computational Problems in Abstract Algebra*, pages 169–184. Pergammon, Oxford, 1970.

[Sim78] C. C. Simms. *Some Group Theoretic Algorithms, Lecture Notes in Math. No. 697.* Springer-Verlag: Heidelberg, Germany, 1978.

[Tod89] S. Toda. On the computational power of PP and $\oplus$P. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 514–519. IEEE Computer Society, Los Angeles, 1989.

[Tse68] G. S. Tseitin. On the complexity of derivations in propositional calculus. In A. O. Slisenko, editor, *Structures in Constructive Mathematics and Mathematical Logic*, pages 115–125. Edinburgh University Press, 1968.

[VV85] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1985.

[Wra76] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3:23–33, 1976.

Appendix A

Binary Decision Diagrams

A.1 Introduction

In this Appendix we briefly describe Ordered Binary Decision Diagrams (OBDDs) and define group action on them. A symmetry argument is given relating canonicity and group action. Work on OBDDs using GAP was described in §4.9. OBDDs are canonical representations of boolean functions as directed acyclic graphs, introduced by Bryant [Bry86]. Many tasks requiring the efficient manipulation of boolean functions have been simplified since the introduction of OBDD methods into fields such as digital system design. The attraction of OBDDs is that there exist tractable algorithms for logical composition of the functions represented as diagrams, while preserving canonicity of the representation.

The diagrams are constructed with respect to a total ordering on the variables. Different orderings yield OBDDs of different sizes, an issue which is of great significance since a good ordering can make the difference between a tractable representation and an intractable one. Knowing symmetries in the function can suggest orderings that are better than random and we illustrated this experimentally in §4.9.

A.2 Definitions

We follow the definitions of [Bry92]. A *Binary Decision Diagram* (BDD) is a rooted directed acyclic graph where: the nonterminal nodes are labelled with variable names from a sequence $\vec{x}$ of variables and have two outgoing arcs, the left and right arcs being

labelled with 0 and 1 respectively; and the terminal nodes are labelled with 0 or 1. For a given assignment $a \in \mathsf{Asg}(\vec{x})$ to the variables, the value yielded by the graph, representing a boolean function over variables $\vec{x}$, is indicated by tracing a branch from the root of the graph, following the arcs corresponding to the value of each variable. The value returned by the function is indicated by which terminal node is arrived at.

An *Ordered* BDD (OBDD) is a BDD with four extra constraints in the context of a given total ordering $\vec{x} = x_1, \ldots, x_n$: i) If a nonterminal labelled with x_i is a child of nonterminal labelled with x_j then $i < j$; ii) there is at least one terminal node labelled 0 or 1 and there are at most two terminal nodes, labelled 0 and 1; iii) no two nonterminals with label x have the same left and right children; iv) no node has two incoming arcs from the same nonterminal.

Notation We use $\mathcal{G}(\phi, \vec{x})$ to represent the (unique) OBDD constructed from formula ϕ using variable ordering $\vec{x}$. There are standard methods for computing OBDDs from arbitrary formulas, see [Bry92]. The above method of tracing an assignment a through the graph to a terminal node corresponds to the evaluation denoted $\mathcal{G}(\phi, \vec{x})(a)$.

Group action Define the action of $S(\vec{x})$ on an OBDD by its action on the variable labels of the vertices, i.e. $\mathcal{G}(\phi, \vec{x})^g$ is the graph of $\mathcal{G}(\phi, \vec{x})$ with x^g replacing each variable label x. Clearly, an OBDD under such a map is still an OBDD. We could extend this action to $W(\vec{x})$ as follows: put $g \in W(\vec{x})$ as δs for $\delta \in \Delta(\vec{x})$ and $s \in S(\vec{x})^{\neg}$ by §2.7.1: apply δ to the graph by interchanging the left and right subtrees of every vertex labelled x for every x moved by δ; apply s as for $S(\vec{x})$ above.

Properties The main point of OBDDs is in identifying 'common subexpressions' of functions. The canonicity property of OBDDs ensures that where such common subexpressions are found relative to an ordering, then they will be exploited. In other words, one never has two nonterminal nodes representing the same function for some residual sequence of variables. We summarise some of the properties for the OBDD construction $\mathcal{G}(\phi, \vec{x})$ in the lemma below.

Lemma A.2.1 The following properties hold:

i). (Correctness) [Bry86] The graph is a correct representation of the formula:

$$(\forall a \in \mathsf{Asg}(\vec{x}))\ \mathcal{G}(\phi, \vec{x})(a) = \phi(a).$$

ii). (Canonicity) [FHS78] For any fixed ordering $\vec{x}$ and wffs $\phi(\vec{x}), \psi(\vec{x})$ then

$$\mathcal{G}(\phi, \vec{x}) = \mathcal{G}(\psi, \vec{x}) \quad \Leftrightarrow \quad \phi \equiv \psi.$$

iii). (Group action) For assignment $a \in \mathsf{Asg}(\vec{x})$ and $g \in S(\vec{x})$ then

$$\mathcal{G}(\phi, \vec{x})^g(a^g) = \mathcal{G}(\phi, \vec{x})(a).$$

A.3 Symmetry argument

We illustrate some of these properties by extracting a simple symmetry argument for OBDDs: the effect of applying $g \in S(\vec{x})$ to an OBDD graph has the same effect as permuting the ordering by g and recomputing the OBDD with the new ordering, if and only if $g \in \Sigma(\phi, S(\vec{x}))$.

Proof We show for any $g \in S(\vec{x})$ that

$$\mathcal{G}(\phi, \vec{x})^g = \mathcal{G}(\phi^g, \vec{x}^g) \tag{A.1}$$

and therefore, by canonicity,

$$\mathcal{G}(\phi, \vec{x})^g = \mathcal{G}(\phi, \vec{x}^g) \quad \Leftrightarrow \quad \phi^g \equiv \phi.$$

We know that $S(\vec{x})$ action preserves the property of being an OBDD, so we know there is some ψ such that

$$\mathcal{G}(\phi, \vec{x})^g = \mathcal{G}(\psi, \vec{x}^g) \tag{A.2}$$

which is equivalent to saying

$$(\forall a \in \mathsf{Asg}(\vec{x}))\ \mathcal{G}(\phi, \vec{x})^g(a^g) = \mathcal{G}(\psi, \vec{x})(a^g).$$

Hence by the valuation property:

$$(\forall a \in \mathsf{Asg}(\vec{x}))\ \mathcal{G}(\phi, \vec{x})(a) = \mathcal{G}(\psi, \vec{x})(a^g)$$

and by correctness:

$$(\forall a \in A(\vec{x}))\ \phi(a) = \psi(a^g).$$

Using $\psi(a^g) = \psi^{g^{-1}}(a)$ then we have $\psi^{g^{-1}} \equiv \phi$ or equivalently $\psi \equiv \phi^g$. So we have determined that the ψ of (A.2) is indeed ϕ^g, establishing (A.1). The result then follows, as indicated, by canonicity.□

What this argument implies is that a permutation g applied to the vertices of an OBDD produces a diagram which defines the same function *if and only if* g is a symmetry of ϕ. This has the following two corollaries. One is the well-known property that if $\phi(\vec{x})$ is invariant under $S(\vec{x})$ then there is only essentially one OBDD representation, a fact that this lemma formalises in slightly more general terms. The other, more general, corollary is that the group $\Sigma(\phi, G)$ partitions the possible diagrams of ϕ with respect to the different orderings. Each partition is closed under action of G on vertex labels. Hence if ϕ has no symmetry then every ordering must give an essentially different diagram.

Appendix B

Computing Closures

B.1 Introduction

This appendix accompanies Chapter 9 in describing the algorithms used by the program 'Brute3' to compute closures and Horn-closures of 3-CNF formulas. Brute3 is written in C and experiments can be run using a range of command-line options. There are also a number of general utilities for manipulating 3-CNFs. A separate document detailing the functionality of the program, as well as a portable version of the code, is available from the author.

At the heart of the program is an efficient satisfiability checker based upon the Davis-Putnam semantic evaluation algorithm, with a number of optimisations to deal with 3-CNFs in particular. The reference to brute force in the naming of the application reflects the main priority in computing closures and Horn-closures for the experiments of Chapter 9, which is the ability to perform very large numbers of satisfiability checks on very large formulas in reasonable time. On the other hand, variable numbers at which these computations are feasible are quite limited because of the $\Theta(n^3)$ factor relating to the numbers of clauses which must be tested in a computation of $\mathsf{Close}(\phi)$. For instance, at 100 variables, computation of $\mathsf{Close}(\phi)$ or $\mathsf{HClose}(\phi)$ requires 1.3 million tests of whether a clause can be added to the formula and this can take several minutes for a reasonable sized formula. In the iterated Horn-closure procedure of §9.4 a further demand on resources is made by the very large size of intermediate formulas, tens or hundreds of thousands of clauses in the experiments conducted. In this case, closure may take considerably longer. Brute3 was designed to be able to perform

these tests with high efficiency, but does not feature sophisticated techniques required for satisfiability checking of formulas with more than 300 or so variables (though it performs well in many cases).

B.2 Closure algorithms

In computing the closure of a 3-CNF or 3-Horn formula $\phi(\vec{x})$, the algorithm embodied in Brute3 essentially makes $\Theta(n^3)$ tests of satisfiability to determine membership of each possible clause in $\mathsf{Close}(\phi)$ where $n = |\vec{x}|$. In general, some preprocessing can replace a considerable proportion of these satisfiability tests with simpler tests. No way has been found to avoid $\Theta(n^3)$ tests of some description. We saw in Theorem 5.5.1 that theoretically we need only perform $O(\log n)$ satisfiability tests, although the argument for this does not appear to give rise in practice to a better algorithm (each of the $O(\log n)$ tests is doing effectively the same amount of, or more, work as a full closure operation–I think). There are in fact three different algorithms which are invoked to exploit formulas of various sizes and hardness: the 'standard', 'quadratic' and 'traverse' algorithms which are described below.

Standard The standard algorithm computes a fixed number, the default is $5v$ where v is the number of variables, of random models a of the formula and uses these as an approximation to the solution space of the formula to reject hopefully large numbers of clauses without having to check inclusion using the satisfiability engine. That is, if we have $\phi(a) = 1$ and $c(a) = 0$ then $\phi \not\models c$ and so clause c can be rejected. Rejection rates for 3-Horn formulas varied between about 12% and 80% in the experiments where the standard algorithm was used.

Quadratic With highly under-constrained formulas such as small to medium size 3-Horn formulas, the quadratic algorithm is invoked. This computes a two-dimensional array of models, attempting to find a model for every distinct pair of literals in which that pair are satisfied. This gives for each candidate clause, up to six quickly accessibly models which can be used to check for rejection. Typical rejection rates are over 90% and when it is practical to use this algorithm, it is very efficient. It is not practical to use this algorithm when the 3-Horn formulas become large: the cost of computing $\Theta(n^2)$

models tends to overshadow the benefit of having them, and the standard algorithm becomes more competitive while the formulas are still relatively under-constrained.

Traverse The traverse algorithm first attempts a full traversal of the solution space of the formula using Brute3's satisfiability engine. Essentially it is trying to extract a DNF version of the formula. If it succeeds in building a DNF representation with no more than some limiting number of clauses then this representation can be used to replace satisfiability checking completely, since it is easy to check membership of clause c in $\mathsf{Close}(\phi)$ directly from the DNF. When a traversal is possible within some reasonable limit on the number of leaves, then this algorithm is very quick to compute the full closure. By default the limit on the number of leaves is also $5v$. Unfortunately it is not usually the case that a full traversal of 3-Horn formulas can be obtained unless they are extremely large. On the other hand the traverse algorithm can produce surprisingly fast responses for a direct closure computation of a 3-CNF when the formula is highly constrained and has a small DNF, and where of course it is feasible to traverse the whole solution space. Critically-constrained formulas are amenable to this approach.

Iterated Horn-closure A typical run of iterated Horn-closure would proceed as follows. While the Horn and non-Horn parts of the formula are small to medium size, up to $30v$ clauses, the quadratic algorithm is invoked to compute the respective closed parts. Beyond this each part is passed to the satisfiability engine to see if it can be represented by a small DNF. If it can, the traversal algorithm is used, if not, the standard algorithm is used. There is usually a slow area in the middle of a run of iterated Horn-closure in which the standard algorithm is being used: the formulas are too big to compute $\Theta(n^2)$ models, and still too under-constrained for it to be practical to find a DNF representation.

Note that these methods involving the computation of large numbers of sieving models are designed to speed up the process for the purposes of experimental analysis over large sample sets. In fact the brute force algorithm which uses satisfiability checking on 3-Horn formulas for each candidate clause is still polynomial overall (Lemma 9.2.1). In general, computing random models of 3-Horn formulas is relatively cheap.

Index